A SOA model, semantic and multimodal, for the discovery and registration of assistance services

Bertha Helena Rodriguez

EDITE ED 130

2013-ENST-0006

Doctorat ParisTech

THÈSE

pour obtenir le grade de docteur délivré par

Télécom ParisTech

Spécialité “ Traitement du Signal et des Images ”

présentée et soutenue publiquement par

B. Helena RODRIGUEZ

le 1 février 2013

Un modèle SOA, multimodal et sémantique et son support pour la découverte et l’enregistrement de services d’assistance

Directeur de thèse : **Isabelle DEMEURE**
Co-encadrement de la thèse : **Jean-Claude MOISSINAC**

Jury

Mme Joëlle Coutaz, Professeur, LIG, Université de Grenoble	**Président**
Mme Laurence Nigay, Professeur, LIG, Université de Grenoble	**Rapporteur**
Mr Yacine Bellik, Assistant-Professor, LIMSI, Université de Paris Sud	**Rapporteur**
Mr Philippe Roose, Professeur, LIUPPA, Université de Pau	**Examinateur**

THÈSE

Télécom ParisTech
Ecole de l’Institut Télécom – membre de ParisTech
46, rue Barrault – 75634 Paris Cedex 13 – Tél. + 33 (0)1 45 81 77 77 – www.telecom-paristech.fr

Acknowledgements

Antes que nada quiero agradecer a Dios, por todo.

Por supuesto, a Isabelle Demeure y Jean Claude Moissinac por su apoyo, guía y paciencia.

A Leonidas, Hernando, Aracely y Berthilda en orden de partida.

A papa, mama, cani, ale, marichi, eloisa, yip, candela, y jimmi, en orden de llegada, por su amor incondicional.

A Benoît, Gladis, Luisfer, Carola, Laurent, Teto, Dianis, Bili, Quim, Titi, Artu, Ira, Evita, Iñaki, Sari, Dianita, Camilo, Sergio, Miguel, Fabian, Lily, Diego, Cucha, Jacqui, Walter, Issame, Flo, Ciro y Alejandro por su amor siempre presente y ejemplar.

A Bernard, Angela, Esteban, Clarita, Christophe, Anne-Marie, Maurice, Nicole, Maria, Myriam, Celine, Clément, Marie-France, Jean-Marc, Sandrine, Antony, Melinda, Mateo, Guillaume, Jason, Mario, Mounira y Harlina, por ser todo mi universo en estas tierras frías.

A Charles y Andy Stanley por enseñarme cada día lo maravilloso que es el camino, la luz, la verdad y la vida.

A Toño, Jaime Pardo Leal, Cesar, Ignacio Abello, Jorge Páramo y Wayne James porque vivo la vida que ya no tienen y agradezco haberlos cruzado.

A Arita, Xavi, Carlos B. Gutierrez, Porfirio Ruiz, Manuel Hernandez, Sergio de Zubiria, Felipe Salamanca, Lucien Sfez, Cesar Garcia, Piedad Bonnett, Alejandro Gonzalez y Laurent Letourmy por inspirarme educándome.

A Catherine Giraud por tomarme de la mano, levantarme y acompañarme semana tras semana con su risa, a Olivier Picaud y Florence Demorand por cuidar de mi salud.

A Hôa Nguyen (tranquilo), Isabel, Angela Anzola, Jean Lefeuvre, Cyril Concolato, Jean-Claude Dufour, Romain Bouqueau, Johnatan Sillan, Deborah Dahl, Kazayuki Ashimura y Piotr Wiechno por su inmensa ayuda y su apoyo.

Y a todos los que se me olvidan (porque quedé muy cansada), mi reconocimiento y mis disculpas.

Paris, Noviembre 2012

Résumé

Les entrées et sorties unimodales dans les systèmes actuels ont atteint une maturité reconnue, avec les applications tactiles ou par les services distribués pour la geo-localisation ou la reconnaissance de la parole, du son ou l'"image. Cependant, l'intégration et l'instanciation de toutes ces modalités, manque d'une gestion intelligente du contexte d'acquisition et de restitution basée sur des notions fortement formalisées mais reflétant le sens commun. Ceci demande un comportement plus dynamique du système avec une approche plus adéquate pour gérer l'environnement de l'utilisateur.

Cependant,la technologie nécessaire pour atteindre un tel objectif n'est pas encore disponible de façon standardisée, tant au niveau des descriptions fonctionnelles des services unimodaux que de leur description sémantique. Ceci est aussi le cas pour les architectures multimodales, où la composante sémantique est produite par chaque projet sans un accord commun pour assurer l'interoperabilité et est souvent limitée au traitement des entrées et sorties ou aux processus de fusion/fission strictement nécessaires au projet.

Pour combler cette lacune, nous proposons une approche sémantique orientée services pour une architecture multimodale générique qui vise à améliorer la description et la découverte des composants de modalité pour les services d'assistance: l'architecture Soa2m.

Cette architecture se veut entièrement focalisée sur la multimodalité et enrichie avec des technologies sémantiques car nous croyons que cette orientation permettra d'enrichir le comportement autonome des applications multimodales, avoir une perception robuste des échanges, et contrôler l'intégration sémantique des interactions homme-machine.

En conséquence, le défi de découverte est adressé à l'aide des outils fournis par le domaine des services web sémantiques. Nous proposons donc, un modèle d'architecture pour fournir des services multimodaux comme une extension de la spécification MMI Architecture and Interfaces du W3C.

Cette extension ajoute des fonctionnalités de découverte et d'enregistrement pour les composants de modalité en utilisant des descriptions sémantiques des processus fournis par ces composants sous la forme de services.

Ceci nous a mené à étendre la recommandation sur plusieurs points: a) une architecture de référence pour les actuels systèmes multimodaux, b) un protocole de communication entre les composants de cette architecture, c) un ensemble d'interfaces entre les différentes composantes de l'architecture, d) un protocole de découverte et d'enregistrement des composants multimodaux, et e) un modèle de données pour la description sémantique des situations (la Soa2m situation) d'utilisation des composants multimodaux à des fins de découverte et d'enregistrement dans les systèmes distribués implementés avec les technologies web.

Abstract

Unimodal inputs and outputs in current systems have become very mature with touch applications or distributed services for geo-localization or speech, audio and image recognition. However, the integration and instantiation of all these modalities, lack of an intelligent management of the acquisition and restitution context, based on highly formalized notions reflecting common sense. This requires a more dynamic behavior of the system with a more appropriate approach to manage the user environment.

However, the technology required to achieve such a goal is not yet available in a standardized manner, both in terms of the functional description of unimodal services and in terms of their semantic description. This is also the case for multimodal architectures, where the semantic management is produced by each project without a common agreement in the field to ensure inter-operability, and it is often limited to the processing of inputs and outputs or fusion / fission mechanisms.

To fill this gap, we propose a semantic service-oriented generic architecture for multimodal systems. This proposal aims to improve the description and the discovery of modality components for assistance services: this is the architecture Soa2m.

This architecture is fully focused on multimodality and it is enriched with semantic technologies because we believe that this approach will enhance the autonomous behavior of multimodal applications, provide a robust perception of the user-system exchanges, and help in the control of the semantic integration of the human-computer interaction.

As a result, the challenge of discovery is addressed using the tools provided by the field of the semantic web services. We propose an architectural model for providing multimodal services as an extension of the MMI Architecture and Interfaces specification of the W3C. This extension adds functionalities for discovering and registration to the modality components using semantic descriptions of the processes provided by these components in the form of services. This led us to extend the recommendation in several aspects: the reference architecture for current multimodal systems, the communication protocol between the components in this architecture, the definition of a set of interfaces between the different components of the architecture, a protocol for discovery and registration of components in large systems, a mechanism to differentially update the descriptions of components in a distributed system and a situational model for the multimodal data (Soa2m the situation) annotation to be used in discovery and registration in distributed systems.

Résumé Etendu

L'un des objectifs initiaux de l'interaction multimodale, est de renforcer la communication naturelle dans les échanges des systèmes informatiques avec les humains. L'un des moyens choisis pour atteindre cet objectif c'est de permettre aux machines de parvenir à un niveau d'intelligence comparable à certains niveaux de l'intelligence humaine. Grâce à ça, la façon dont les humains interagissent avec eux est censée s'améliorer.

Selon certaines tendances actuelles en intelligence artificielle, il est possible aujourd'hui d'arrêter de traiter les machines comme des machines et en tant que développeurs, commencer à programmer les machines et leur transmettre nos connaissances (et la sémantique de nos faits et gestes), comme s'il s'agissait d'entités sociales telles que nous.

Aujourd'hui, même s'il y a beaucoup de travail à faire pour décrire l'expérience de nos structures cognitives à un système multimodal, beaucoup a déjà été accompli d'un point de vue pratique.

Les entrées et sorties unimodales dans les systèmes actuels ont atteint une maturité reconnue, dans des systèmes comme Siri ou les applications tactiles, ou par les services distribués («in the cloud») pour la reconnaissance de la parole, le son et l''image. Aussi, l'expérience sensorielle des événements multimodaux dans la vie courante est aujourd'hui plus riche et largement documentée (par exemple par le crowd media des réseaux sociaux).

Les données provenant de différents capteurs -sensors en anglais- sont capturées et stockées, et l'interaction unimodale est monitorée en outre, par des processus d'apprentissage statistique. Cependant, l'intégration et l'instanciation de toute cette information, riche en modalités, manque d'une gestion intelligente du contexte d'acquisition et de restitution basée sur des notions fortement formalisées mais provenant du sens commun.

Nous croyons que ce type de sémantique permettra d'enrichir le comportement autonome des applications multimodales, une perception robuste des échanges, et le contrôle et l'intégration sémantique des interactions homme-machine.

Mais nous devons commencer par le commencement. Et pour les systèmes déployés et produits avec des technologies web, le commencement peut être l'annotation de cette connaissance, sa découverte et de son enregistrement pour être mise à disposition du système.

Donc, une des motivations derrière ce travail de thèse, est la proposition des outils pour les mécanismes de découverte dans une architecture orientée services. Cette architecture se veut entièrement focalisée sur la multimodalité et enrichie avec des technologies sémantiques.

Cependant, la technologie nécessaire pour atteindre un tel objectif n'est pas encore disponible de manière standardisée, tant au niveau des services dans la toile que au niveau de son enrichissement sémantique. Ce qui est vrai pour le domaine des Architectures de systèmes multimodaux, à savoir, le manque de bonnes pratiques de conception standardisées qui pourraient conduire à une théorie et une application plus automatique et interoperable de certains principes; est également vrai pour les services web sémantiques.

Ce manque de technologie va probablement disparaître dans les prochaines années et cette thèse vise à développer un ensemble d'outils de base, qui pourront servir comme un point de départ pour résoudre cette limitation et avancer vers un processus de standardisation.

Avec cet objectif, nous nous sommes inspirés des arts vivants, et plus précisément de l'expérience que nous avons acquis dans la construction des installations interactives multi-sensorielles et *in-situ* pour encadrer la création de quelques outils pouvant aider à construire (en utilisant les ressources fournies par le cloud) la prochaine génération d'applications intelligentes et ludiques, basées sur des expériences multimodales riches.

Historiquement, les systèmes multimodaux ont été mis en œuvre dans des environnements stables et bien connus (cockpits, salles de conférence, salles de classe, kiosques publicitaires) pour un nombre réduit et bien connu d'utilisateurs. Leur complexité de mise en œuvre a fait que peu d'expériences ont été développées pour une utilisation temps réel, dans une situation de mobilité des utilisateurs dans des contextes multiples et non contrôlés, ou avec des composants distribués. Néanmoins celles-ci sont les conditions actuelles de consommation massive d'applications surtout en raison de l'évolution du web mobile. Et nous pensons qu'il est possible de présumer que ces applications deviendront de plus en plus multimodales.

Car au même temps, les développeurs web commencent à relever le défi multimodal, principalement grâce à la disruption crée par le marché des smartphones et les applications proposant la reconnaissance vocale et tactile, les API avec widgets proposant des services distribués d'interface utilisateur, par exemple Google Maps, GoogleVoice, GoogleEarth, Tropo, Face.com, TouchATag, etc. Par conséquent, la communauté de développeurs web est progressivement confrontée au problème de l'intégration des modalités avec des ressources d'entrée ou sortie dans les réseaux à grande échelle.

Ceci demande un comportement plus dynamique du logiciel avec une approche plus adéquate pour gérer l'environnement de l'utilisateur.

Dans ce contexte, la découverte et la sélection des modalités d'interaction dans les applications riches en media devient un nouvel horizon de travail donnant de nouveaux défis pour les systèmes multimodaux avec une conception centrée utilisateur et produits avec les technologies web.

La principale lacune dans le réseau à grande échelle (et ubiquitaire) actuel est le manque de moyens pour une application pour découvrir les web services et les applications disponibles dans un espace et un réseau donnés, surtout si cette découverte se veut basée sur des besoins exprimés avec une sémantique de très haut niveau. Ce problème est partagé aussi par les services de modalité distribués, qui peuvent être utilisés dans des applications multimodales.

Pour combler cette lacune, nous proposons une approche sémantique orientée services pour une architecture multimodale générique qui vise à améliorer la description et la découverte des composants de modalité pour les services d'assistance: l'architecture Soa2m.

Pour nous, ce défi de découverte peut être adressé à l'aide des outils fournis par le domaine des services web sémantiques. Pour cette raison, nous proposons un modèle d'architecture pour fournir des services multimodaux comme une extension de la spécification MMI Architecture and Interfaces du W3C. Cette extension ajoute des fonctionnalités de découverte et d'enregistrement des composants de modalité en utilisant des descriptions sémantiques des processus fournis par ces composants sous la forme de services.

D'autre part, l'annotation sémantique des composants et des processus fournis par des services d'interaction utilisateur, est également un domaine de recherche très prometteur. Néanmoins, il n'y a pas beaucoup de références dans ce domaine, étant donné que la plupart des travaux sur l'alignement sémantique est orientée vers les systèmes d'information des entreprises ou les loisirs.

L'interaction homme-machine, ou des questions spécifiques au traitement des interactions multimodales sont absents dans ces travaux qui se concentrent principalement dans les données commerciales ou culturelles et l'annotation des médias de divertissement.

Finalement, dans les systèmes ubiquitaires avec une approche orientée services, la multimodalité est traitée comme un problème de gestion des entrées ou des sorties, de la reconnaissance ou de la présentation des modalités, en laissant de côté la proposition d'une architecture générique nécessaire pour les développeurs pour assurer la découverte, le monitorage et la coordination en temps réel, ou adaptée au contexte d'usage des modalités.

Par conséquent, dans les cas concrets d'implémentation, les applications multimodales ou multimédia sont composées manuellement en utilisant les API intégrées d'une manière ad-hoc.

L'architecture MMI répond à ce manque de cadre de référence, mais pour le moment, elle ne couvre pas la gestion du cycle de vie des modalités, ou l'adaptation au contexte qui utilise la découverte, l'enregistrement des composants et sa mise à jour.

Avec des informations sur le contexte, sur les besoins non-fonctionnels et sur les caractéristiques multimodales, tout ceci décrit par des technologies sémantiques, le comportement d'une application pourrait être plus dynamique. Ceci peut être fait par la création des services intelligents décrits à haut niveau associés à un espace particulier et une situation d'interaction particulière en utilisant une architecture adaptée pour instancier et gérer des telles services. Et cette architecture doit effectuer la découverte et l'enregistrement des processus adaptés à la situation concrète.

Pour résumer, dans Soa2m les processus de découverte et d'enregistrement sont adressés par trois moyens: une architecture qui étend une recommandation du W3C (MMI Architecture and Interfaces); trois formats de description des services basées sur des Tags, des annotations sémantiques et des documents WSDL enrichis avec des ontologies, et finalement, un modèle de données pour l'enregistrement des services de modalité basés sur un ensemble de propriétés multimodales et la description sémantique de la situation d'utilisation multimodale.

Ainsi, une dernière motivation de ce travail de thèse, est de contribuer à la normalisation de:

- une architecture de référence pour les actuels systèmes multimodaux
- un protocole de communication entre les composants de cette architecture

- un ensemble d'interfaces entre les différentes composantes de l'architecture
- un protocole de découverte et d'enregistrement des composants pour les systèmes implementés avec les technologies web.
- un mécanisme de mise à jour différentielle de descriptions de composants dans un système distribué multimodal
- un modèle de données pour la description sémantique des composants multimodaux pour la découverte et l'enregistrement dans des systèmes distribués

La thèse est structurée en trois chapitres et un certain nombre d'annexes.

L'introduction générale présente la motivation, un aperçu du problème et la structure du document. Il présente également la méthodologie de recherche que nous avons appliqué.

Chapitre 1 - Systèmes multimodaux, s'occupe de la définition des systèmes multimodaux à la lumière des nouveaux modes d'interaction et des nouveaux champs de recherche ouverts dans les dix dernières années par les changements dans le domaine technique et dans le marché des technologies de l'information. Nous définissons d'abord certains termes. Ces concepts sont: *système capteur*, *système effecteur*, *mode*, *modalité*, *medium*, *fusion* et *fission*.

Deuxièmement, nous étudions la description empirique de ce qu'un système multimodal pourrait être aujourd'hui, basée sur l'analyse des outils existants et les mécanismes actuellement utilisés pour relever le défi multimodal. Au-delà des caractéristiques communes, nous explorons quelles peuvent être les potentielles caractéristiques d'un système multimodal conçu aujourd'hui:

- Caractéristiques liées au comportement de l'utilisateur et du système concernant la gestion des sessions, la gestion des événements, la stratégie de dialogue ou le processus décisionnel.
- Caractéristiques liées à la description de certains participants d'interaction: les dispositifs, les utilisateurs et le domaine de l'interaction.
- Caractéristiques liées au contexte général d'exécution du système: la situation d'utilisation, la situation temporelle et la situation spatio-temporelle.

Ensuite, nous discutons un possible cadre général pour un système multimodal du point de vue de la classification de ces systèmes. L'objectif est la détection des éventuels blocs fonctionnels pour une architecture générique de référence.

Nous présentons quelques systèmes focalisés sur ce cadre de référence pour délimiter la portée de la notion de système multimodal dans nos recherches et dans le but d'expliquer ce qu'est un système multimodal aujourd'hui. Ceci est réalisé en vérifiant dans des implémentations concrètes et leurs architectures multimodales combien des bloc de construction sont présents et comment ils sont conçus. En passant en revue ces implémentations réelles nous avons confronté l'initiative multipartite du W3C (industriels, gouvernements et chercheurs internationaux) de recommandation d'une architecture multimodale de référence pour analyser sa portée.

Avec cet objectif, tout d'abord, nous présentons ce que nous avons appelé «MMI Framework & Architecture» qui englobe deux initiatives de spécification du W3C: la spécification MMI Architecture and Interfaces, et la recommandation Multimodal Interaction Framework.

En deuxième lieu, nous avons étudié les architectures multimodales les plus pertinents selon les exigences techniques fournies et, enfin, nous évaluons «MMI Framework & Architecture» en opposition avec les implémentations concrètes étudiées. Ceci est produit avec un échantillon de seize architectures sélectionnées après une analyse préalable d'un ensemble plus vaste de cent implémentations multimodales. Les critères de sélection ont été la quantité d'information fournie par les auteurs sur les aspects architecturaux de la mise en œuvre, son exhaustivité et sa représentativité de plusieurs domaines de recherche que nous avons trouvé pertinents.

Trois niveaux d'analyse ont été fournis: tout d'abord, une description fonctionnelle de l'architecture, d'autre part, une comparaison visuelle des blocs fonctionnels exprimés par des codes de couleur et enfin, l'analyse de quatre séries de critères regroupant les caractéristiques énumérées ci-dessus.

Pour chaque architecture étudiée, un petit nombre de faits intéressants ont été mis en évidence dans la perspective de l'enrichissement d'une architecture de référence envisagée pour des fins de standardisation et d'interopérabilité.

Les résultats sont deux groupes de tendances qui se dégagent du choix des blocs fonctionnels de cet échantillon historique d'architectures multimodales. En analysant chaque approche à la lumière de nos critères, nous faisons ressortir les contributions possibles de chacune selon l'objectif d'une architecture multimodal sensible au contexte, pour les réseaux étendus comme l'Internet. Cette analyse empirique et qualitative de l'état de l'art présente le positionnement du projet Soa2m, qui est décrit dans la deuxième partie de cette thèse.

Chapitre 2 - Le projet Soa2m est l'effort d'aborder ces thèmes émergentes dans la recherche multimodale dans le cadre du travail sur les normes ouvertes du Web et plus précisément la participation et la contribution à l'activité du W3C dans le groupe de travail en interaction multimodale.

Dans ce chapitre, nous présentons le projet, décrivons la proposition et expliquons les implémentations actuelles au sein du projet Soa2m.

Alors que le premier chapitre a présenté un examen approfondi des choix de conception pour les architectures multimodales sur une longue période de temps, nous avons gagné dans la compréhension de la façon dont les architectures multimodales systèmes a évolué. Ceci a été important pour examiner plus en détail ce qui pourrait être ajouté à une hypothétique architecture de référence résultant de standards ouverts.

Une généralisation finale nous a permis de construire les hypothèses architecturales qui composent le noyau de l'approche du projet Soa2m: la proposition d'un ensemble d'extensions de la recommandation «MMI Framework & Architecture» .

Soa2m vise à fournir à la norme du W3C le support architectural des exigences émergents de mise en œuvre comme: l'adaptation intelligente, dynamique et plastique de l'interface utilisateur et du cycle d'interaction avec l'utilisation des meilleures ressources disponibles pour communiquer un message et pour générer une expérience utilisateur sur plusieurs modes, modalités et média.

Alors, dans ce chapitre, tout d'abord nous présentons l'architecture Soa2m et ses exigences, en second lieu, nous décrivons le support sémantique prévu dans l'architecture pour améliorer la découverte et l'enregistrement des composants dynamiques et, troisièmement, nous décrivons la mise en œuvre de certains outils pour l'annotation sémantique des services de modalité.

L'approche architecturale de Soa2m, est conçue pour répondre à l'hypothèse de travail suivante: «il devrait être possible de construire un système apportant des services à chaque utilisateur avec l'utilisation des meilleures ressources disponibles décrites sémantiquement, indépendamment du canal de communication, du mode d'acheminement, des média ou du dispositif. »

Après la sélection d'une architecture de référence comme point de départ, à savoir la spécification «MMI Framework & Architecture», l'analyse de cette solution standardisée nous a amené à proposer les extensions nécessaires pour répondre à nos besoins.

Ces extensions sont présentées à partir de quatre points de vue: la vue logique, la vue des processus, la vue de mise en œuvre et, enfin, la vue du déploiement physique. Ensuite, le choix du style d'architecture pour Soa2m et ses propositions détaillées est confronté à une série d'exigences fonctionnelles et non fonctionnelles. Les extensions offertes par l'architecture Soa2m proposent des améliorations à un modèle existant. En tant que modèle, elle est évaluée non seulement en fonction de ses besoins fonctionnels (qui peut évoluer au fil du temps), mais surtout en termes des besoins non-fonctionnels qui garantissent sa capacité à être adoptée par une large communauté de développeurs.

En conséquence, nous montrons que l'architecture Soa2m répond également à ces exigences non fonctionnelles, parce qu'elle a été conçue avec cet objectif explicite, en se concentrant sur trois points de vue: les exigences non fonctionnelles au moment de la conception, les exigences non fonctionnelles au moment de l'exécution et enfin, les exigences non fonctionnelles affectant l'interaction de l'utilisateur avec le système.

Le chapitre complète également la description de l'architecture orientée services proposée par Soa2m avec la présentation de son approche sémantique. L'objectif est de présenter cette partie complémentaire de l'approche, à savoir l'enrichissement des services avec des annotations de métadonnées provenant de la recherche sur les Web services. Ces procédures d'annotation sont conçues pour améliorer la découverte sémantique, l'enregistrement, la mise à jour et la gestion des services d'un système multimodal créé en accord avec la proposition d'architecture Soa2m.

Comme l'interaction multimodale dans les réseaux étendus se déroule dans des conditions très dynamiques et imprévisibles, une découverte réactive et proactive est nécessaire afin de répondre à toute mise à jour du système. Le système doit également «comprendre» ses états internes et externes, le temps, les buts, les entités (les dispositifs, les services, les modalités, les médias), les participants (des utilisateurs ou des systèmes), les propriétés, les catégories et les relations entre eux.

Ce chapitre présente donc l'approche sémantique du Soa2m concernant les métadonnées nécessaires pour les processus de découverte et enregistrement des données multimodales. Ceci est fait à partir de trois perspectives: la perspective des exigences, la perspective des techniques d'annotation et enfin la perspective du modèle de données.

Tout d'abord, nous présentons les exigences relatives à la gestion des services multimodaux, en ce qui concerne l'annonce des services, leur découverte et leur enregistrement. Cela nous a permis de détailler certaines des responsabilités fonctionnelles pour l'architecture Soa2m, et cela nous a donné un cadre pour la description sémantique. Ici, nous avons présenté deux propositions pour un protocole de découverte et d'annonce pour des systèmes multimodaux et deux extensions de la spécification du W3C afin de supporter la gestion dynamique des composants.

Puis, dans un deuxième temps, nous avons décrit les services dans Soa2m comme des processus qui sont consommés selon des accords fonctionnels et non fonctionnels d'un contrat de service suivant une intention générique. La description est orientée vers le support du mécanisme de découverte et d'enregistrement, car la principale contribution de cette thèse est autour de cette question; à savoir: comment gérer la disponibilité dynamique des services de modalité à travers l'architecture Soa2m.

Or nous avons présenté le mécanisme d'annotation avec les technologies du web sémantique existants, laissant pour la dernière partie du chapitre l'explication sur le modèle de données utilisé dans ce processus d'annotation. En d'autres termes, ce chapitre aborde d'abord les méthodes d'annotation sémantique en laissant pour plus tard la discussion sur le modèle de données.

Nous avons présenté trois formats et méthodes d'annotation (des structures de données, des manifestes sémantiques, des extensions aux documents WSDL) adaptés aux différents besoins. Ces procédures d'annotation de Soa2m envisagent l'utilisation intelligente des technologies existantes pour accomplir notre but d'annoter non seulement le comportement fonctionnel, mais aussi les conditions contextuelles pour une utilisation synergique[1] des interfaces multimodales. Ces procédures répondent à trois exigences (l'annonce, la découverte et l'enregistrement) et sont compatibles avec la perspective architectural de Soa2m.

Troisièmement, nous avons décrit comment les entités multimodales sont modélisés dans Soa2m afin de soutenir le mécanisme de prise de décision pour l'adaptation du système au contexte d'utilisation. Nous avons présenté les ontologies produites dans Soa2m pour annoter les situation multimodales.

Ensuite nous avons présenté le modèle de données pour être utilisé en association avec nos trois méthodes d'annotation. La proposition est fondée sur la prémisse que la description des services couvre deux approches: le fonctionnement des services et l'utilisation du service. Alors que le premier concerne les opérations et les types des données, la seconde est guidée par la notion de situation. En conséquence, nous avons proposé un glossaire d'annotation multimodale pour l'annonce et de la découverte et un modèle d'ontologie pour la découverte et l'enregistrement des services de modalité.

Ce dernier est une ontologie à facettes, qui classe la situation en quatre couches de connaissance en fonction des intérêts de l'agent bénéficiaire de la description: la facette sensorielle liste des choses dont le destinataire aurait besoin, la facette du comportement décrit les comportements des agents et des choses, la facette sémantique décrit la dimension sociale (le domaine) et le sens des concepts utilisés par les agents ou les choses, et finalement, l'aspect intentionnel qui décrit l'intention annoncée des agents ou des choses. Avec ces facettes, la situation est décrite selon le point de vue et les intérêts de l'agent final qui va utiliser cette description.

D'autre part, la situation Soa2m est également décrite avec une certaine granularité: la granularité ponctuelle se concentre sur l'énumération, la granularité relationnelle se concentre sur les connexions et la granularité symbolique, est orientée vers l'utilisation de la connaissance analogique[2]. Avec ces granularités, la situation est expliquée selon un point de vue précis ou avec un ensemble des couches complémentaires.

1 Synergie signifie «travailler ensemble». et c'est le terme utilisé pour décrire deux ou plusieurs entités qui fonctionnent ensemble pour produire un résultat qui ne pourraient être obtenu par chacune indépendamment.

2 C'est un concept qui porte sur des réalités essentiellement diverses, mais qui, cependant ont entre elles une certaine relation. C'est une notion qui s'applique à des sujets qui sont perçus ni totalement identiques ni totalement différents: ils se ressemblent dans un aspect précis mais aussi ils possèdent au moins une différence certaine. L'analogie est l'une des trois formes de raisonnement, avec la déduction et l'induction. Elle consiste à établir des rapports entre des domaines différents de réalité, soit pour illustrer un attribut, soit pour découvrir un aspect inconnu.

Enfin, la situation Soa2m a une description en trois couches. Le couche *Who* (qui) décrit les agents participant à la description. La couche *How* (comment), décrit les processus qui affectent la situation. Enfin, la couche *Which* (quoi) décrit les entités nécessaires à la situation: *Where* (lieu), *When* (attribut temporal) et *What* (l'entité).

Pour terminer ce chapitre, nous avons introduit cinq implémentations de bibliothèques, comme une contribution à l'adoption des spécifications ouvertes pour la multimodalité:

- La Bibliothèque Soa2m sémantique des services est un outil pour l'annotation des services de composants de modalité avec les technologies du web sémantique
- La Bibliothèque Soa2m MMI Lib est un outil pour la mise en œuvre du protocole de communication MMI.
- La bibliothèque d'architecture Soa2m est un outil pour la mise en œuvre d'architectures sensibles au contexte et possédant une sémantique intelligente qui étendent la norme «MMI Framework & Architecture». La bibliothèque permet aussi d'ajouter à la description l'ontologie Soa2m de situation. Ceci est proposé pour les systèmes multimodaux ayant besoin des connaissances sur le contexte d'utilisation.
- Un modèle de l'utilisateur et l'ontologie des modes pour décrire des phénomènes multimodaux.

Chapitre 3 - Contribution fournit les principales contributions de notre recherche à partir d'un point de vue théorique (l'approche conceptuelle) et d'un point de vue technologique. Le chapitre se termine par quelques remarques finales au sujet de notre contribution à l'effort de standardisation du W3C et des lignes de recherche qui restent ouvertes. En guise de résumé nous pouvons lister les plus importants points de contribution à la spécification «MMI Framework & Architecture»:

- Une mise en œuvre complète de l'architecture MMI et Interfaces (avec 7 composants complexes de modalité et 1 manager d'interaction globale) a été analysée et décrite par un long et très complet rapport d'implémentation. Cette mise en oeuvre a été fournie comme l'une des quatre mises en oeuvre qui permettent de valider la norme.
- La coordination du nouveau sous-groupe «discovery & registration» du groupe de travail en interaction multimodale, qui a conduit à la proposition d'un ensemble de cas d'utilisation et de leurs besoins techniques. Trois des cas d'utilisation, et l'ensemble des besoins sont sortis de la proposition du projet Soa2m et ont été validés par le processus de recommandation du W3C.
- La proposition de quatre contributions spécifiques pour la prochaine norme ouverte qui sera publiée par le Groupe de travail MMI: deux extensions architecturales afin de soutenir la gestion de l'état des composants, deux nouveaux événements pour répondre aux besoins de gestion de découverte, deux nouveaux protocoles pour la découverte des modalités, l'annonce des composants de modalité pour l'amorçage; et la création d'un vocabulaire commun et interopérable de compétences génériques pour permettre une première découverte grossière des modalités dans les réseaux étendus.
- La promotion et la documentation de l'architecture MMI en anglais et français à travers l'Internet et les médias sociaux, comme Wikipedia.

Table of Contents

GENERAL INTRODUCTION

Motivation

Historically, multimodal systems were implemented in stable and well-known environments (cockpits, conference rooms, class-rooms, advertising kiosks). Their complexity demanded laboratory-like implementation and very few experiences were developed for real-time use, user mobility over multiple and uncontrolled context or component distribution.

In the meantime, web developers begin to raise the multimodal challenge, mostly thanks to the smart phone market disruption produced by applications proposing speech recognition and touch, and API's provided by e.g. the Google Maps widget, Voice, Earth. Therefore, the web developer's community is progressively confronted with the problem of modality integration in large-scale networks.

The increasing amount of user produced and annotated media and interface services will require a more dynamic software behavior with a more adequate approach to handle the user environment.

In this context, on one side, modality discovery and selection for rich media applications becomes a new working horizon giving new challenges for multimodal systems, user-centric design and web research.

Thus, the motivation behind this thesis work is the proposal of discovery and registering tools for services oriented architectures fully focused on multimodality and enhanced with semantic technologies, in order to contribute to the standardization of:

- an architecture of reference for current multimodal systems
- a protocol of communication between components in this architecture
- a set of interfaces between the different components of the architecture
- a protocol of discovery and registering of components for large scale systems
- a mechanism of differential update of component descriptions in a multimodal distributed system
- a data model for the semantic description of multimodal components for discovery and registering in large scale and distributed systems

Problem Overview

The major gap in the current ubiquitous web is the lack of means for an application to discover services and applications available in a given space and network based on the high-level semantics of its needs. This problem is shared by the distributed modality components that can be used in multimodal applications. To address this gap we propose a Service Oriented architectural approach that aims to enhance the description and discovery of Modality Components for assistance services with the advertisement of semantic descriptions: the Soa2m architecture.

For us, the discovery challenge can be addressed using tools provided by the field of the semantic services. For this reason we propose an architecture model to provide multimodal services as an extension to the MMI Runtime Framework and the MMI Architecture Recommendation. This extension covers the discovery and register of multimodal features using semantic descriptions of the processes provided by the MMI Components.

On the other side, the semantic annotation of components and processes providing user interaction services is also a very promising field of research. Nevertheless, there are not many references in this field, since most of the work about semantic matchmaking is oriented to enterprise or media information systems. Human-computer behavior, interaction processing or multimodal specific issues are absent in these works that focus mostly in business data or cultural and entertainment media annotation. In the case of ubiquitous Service Oriented Architecture systems, multimodality is treated as an input or output problem of recognition or modality management, leaving aside the problem of a generic architecture proposal needed by web developers for the discovery, monitoring and coordination of modalities in real-time and with context-awareness.

Consequently, in real-life consumption cases, multimodal media applications are manually composed and shared via web APIs and embedded web technologies in a 'hacked' and ad-hoc way. The MMI Architecture addresses this lack of frame of reference but for the moment, it does not cover context management or the runtime framework lifecycle that includes the discovery and register of components. With this context and modalities handling, a more dynamic application behavior based on semantic descriptions of non-functional goals and multimodal features can be achieved. This can be made through the creation of intelligent high-level services associated to a particular space and a particular interaction situation using an adapted architecture. And such architecture has to perform discovery and register processes adapted to the concrete real-life situations.

To sum up, in Soa2m discovery and registering processes are addressed by three means: an Architecture extending the Multimodal Architecture and Interfaces W3C recommendation; three formats of service description based on tagging, semantic annotations and an extended WSDL enriched with ontologies; and finally, a data model for the registering of Modality Component services based on multimodal properties and the semantic description of the multimodal situation of use.

Outline

The thesis is structured in three chapters and a number of appendices.

The General Introduction outlines the motivation, the problem overview and the structure of the document. It also presents the research methodology we applied.

Chapter 1 - Multimodal Systems, define multimodal systems on the light of the new modes of interaction and research fields opened by the last ten years of changes in the technical realm and in the information technology market. First we will define some terms. These concepts are *sensor system, effector system, mode, modality, medium, fusion* and *fission*. Secondly we study a more empirical description of what a *multimodal system* could be today, based on the analysis of existing tools and the mechanisms currently used to address the multimodal challenge. Beyond the common characteristics, we explore what other characteristics a *multimodal system* designed today can **potentially** encompass:

1. Characteristics related to the system's and user's behavior regarding the session management, the event handling, the dialog strategy or the decision making process.
2. Characteristics related to the description of some interaction participants : devices, users and domain data.
3. Characteristics related to the general delivery context of the system: the usage situation, the temporal situation and the spatial situation.

Then we discuss an intended frame of reference for a multimodal system as a standpoint for the classification of these systems and for the detection of possible architectural building blocks for a generic Architecture of Reference. We present some systems through the lenses of this frame of reference in order to understand the scope of the *multimodal system* notion in our research and in order to explain what a «multimodal» system is today. This is realized by checking in real multimodal implementations and architectures how many of these building blocks are present and how they are designed. By passing across real implementations of multimodal architectures we confront the multisided initiative of the W3C (international industrials and researchers) for a multimodal architecture of reference with the various experiences of implementation and theoretical proposals for multimodal architectures and some principles of architectural design for multimodal systems.

With this goal, first, we present the Multimodal Runtime Framework & the Multimodal Architecture and Interfaces recommendations; second we discuss the most relevant multimodal architectures according to the requirements given and finally, we evaluate the Multimodal Runtime Framework model in opposition to the concrete implementations studied. This is made with a sample of sixteen architectures selected from a previous analysis of a larger set of multimodal implementations[3]. The selection criteria has been the amount of information provided by the authors about the architectural facets of the implementation, its completeness and its representativeness of some domains of research that we found relevants.

Three levels of analysis were provided: first, a functional description of the architecture, second, a visual comparison of the functional blocks expressed with color codes and finally, the analysis of four sets of criteria gathering the characteristics listed above.

For each architecture studied, a few number of interesting facts were highlighted with the perspective of the enrichment of an architecture of reference envisioned for the purpose of standardization and interoperability.

As a result, we present two groups of trends that emerge from the choice of functional blocks of this historical sample of multimodal architectures. By analyzing each approach under the light of our generic criteria, we bring out the possible contributions of each approach against the goal of a context-aware Multimodal Architecture of Reference, for large-scale networks like the Internet. This empirical and qualitative analysis of the state of the art introduces the positioning of the Soa2m project, which is described in the second part of this thesis.

3 See Appendix 1 Multimodal Architectures Timeline

Chapter 2 - The Soa2m Project is the effort to address this emergent topics in the multimodal research, framed by the work on open web standards and more precisely the participation and contribution to the activity of the W3C's Multimodal Working Group.

In this chapter we present the project, describe the proposal and explain the current implementations.

While the first chapter presented an in-depth examination of the design choices for multimodal architectures over a long period of time we gained understanding of how the multimodal systems architectures evolved; what might become important to look at more extensively; and what might be added to a hypothetical architecture of reference resulting from open standards.

This led us to a final generalization to build some architectural hypotheses that compose the core of the Soa2m project approach: the proposal of a set of extensions to the MMI Framework & Architectures.

Soa2m seeks to provide with the W3C's standard, the architectural support of emerging implementation requirements like the intelligent, dynamic and plastic adaptation of the user interface and the interaction cycle with the use of the best resources available to communicate a message and to generate a user experience in multiple *modes*, *modalities* and *media*.

In this chapter, then, first, we present the Soa2m architecture and its requirements, second, we describe the semantic support envisioned in the architecture to enhance discovery and registration of dynamic components and third, we describe the implementation of some tools for the semantic annotation of these services.

The Soa2m architectural approach, is designed to address the following working hypothesis: «it could be possible to build a system bringing services to each user with the use of the semantically best resources available, independently of the communication channel, the *mode* of restitution, the *media* or the device.»

After the selection of a reference architecture as a starting point, namely, the MMI Framework & Architecture Specification, the analysis of this standardized solution lead us to propose some extensions required to address our need.

These extensions are presented from four points of view: a logic view of the proposal, a process view of the proposal, its implementation point of view and finally the physical view of the proposed deployment. Then, the selection of the Soa2m architecture style and its previously detailed proposals is confronted to a series of functional and non-functional requirements. The extensions offered by the Soa2m architecture are enhancements to an existent model. As a model, it is evaluated not only in terms of its functional requirements (that can evolve over time) but mostly in terms of non-functional requirements that ensures its capability to be adopted by a large community of developers.

As a result, we show that the Soa2m architecture also responds to these non-functional requirements and was designed with this explicit goal, focusing on three perspectives: the non-functional requirements at design-time, the non-functional requirements at run-time and finally, the non-functional requirements affecting the user interaction with the system or the designer interaction with the implementation proposal.

This chapter also complete this description with the presentation of the semantic approach of the Soa2m services-oriented architecture. The goal is to present a complementary part of the approach, namely, the enrichment of services with metadata annotations coming from the semantic web services community designed to enhance the discovery, registration, update and management of services of a *multimodal system* created according the Soa2m architecture proposal.

As multimodal interaction in large-networks takes place in highly dynamic and unpredictable conditions, reactive and proactive discovery is needed in order to respond to any incoming update of the system. The system also needs to "understand" internal and external states, time, goals, entities (devices, services, *modalities*, *media*), participants (users or systems), properties, categories and relations between them.

This chapter presents also the Soa2m semantic approach concerning the metadata needed for multimodal discovery processes and data in multimodal systems. This is made from three perspectives: the requirements perspective, the annotation techniques perspective and finally the data model perspective.

First, we present the requirements for the management of multimodal services, regarding the advertisement of services, their discovery and the registration of services. This will let us to detail some functional responsibilities for the multimodal Soa2m architecture, and also it will give us a context for the description of multimodal processes and data. Here we introduced two proposals for a discovery and advertisement protocols for multimodal systems and two extensions to the W3C specification in order to support the dynamic management of components.

Then, in a second step, we will explain the description of services in Soa2m as processes representing a service consumption contract or an intent. The description will be oriented to the discovery and registering mechanism, given that the main contribution of this thesis is around this issue, namely: how to handle the dynamic availability of Modality Component Services through the Soa2m architecture.

In this section we present the annotation mechanism proposed using the existing semantic web technologies, letting for the final part of the chapter the explanation about the data model used in this annotation process. In other words, this chapter will address first the methods of semantic annotation leaving aside the discussion about the data model for a moment. Here we presented three formats and methods of annotation (data structures, semantic manifests, extended web service description documents) adapted to different needs. These Soa2m annotation procedures , address the intelligent use of some existent technologies to fulfill our purpose of annotating not only functional behavior but also contextual and usage conditions in a synergic way. These procedures respond to our three requirements (advertisement, discovery, registering) and are compatible with the Soa2m architectural perspective.

Third, we describe how multimodal entities are modeled in Soa2m in order to support the decision making mechanism for the context-awareness of the system. In this section we will introduce the Soa2m ontologies and the proposal of a multimodal situation data model.

We presented the data model to be used in association with these annotation methods. The proposal is based on the premise that services description covers two approaches: the service functioning and the service usage. While the first approach concern processes, the second is guided by the notion of situation. In result, we proposed a multimodal annotation glossary for advertisement and discovery and an ontology knowledge model for discovery and registering.

The latter is a facetted ontology, that classifies the situation into four layers of knowledge -the facets- depending on the interests of the agent recipient of the description: the sensorial facet lists things needed by the recipient, the behavioral facet describes behaviors of agents and things, the semantic facet describes the social (domain) meaning of agents or things and the intentional facet that describes the advertised intention of agents or things. With these facets, the situation is explained according to the perspective and interests of the final agent that will use it.

On the other hand, the Soa2m situation is also described with a given granularity: the punctual granularity focuses on enumeration, the relational granularity focuses on the connections and the symbolic granularity, is oriented to the use of analogical knowledge. With these granularities, the situation is explained according to a precise point of view.

Finally, the Soa2m situation has a description on three layers.The Who layer, describes the agents participating in the description. The How layer, describes the processes in the situation. Finally the Which layer, describes the entities needed for the situation: where (a place), when (a temporal region) and what (an entity).

To finish the chapter we introduce the five implementations of libraries as a contribution to the adoption of multimodal standards:

- The Soa2m Semantic Services Library as a tool for the annotation of Modality Component services with semantic web technologies
- the Soa2m MMI Lib as a tool for the implementation of the MMI protocol of communication.
- the Soa2m Architecture library as a tool for the implementation of context-aware and semantically intelligent architectures extending the MMI Architecture and Interfaces recommendation.
- the Soa2m Situation ontology to enrich multimodal systems with context knowledge.
- the User model and the Mode ontology to describe multimodal phenomena.

Chapter 3 - Contribution provides the major contributions of our research from a theoretical point of view (the conceptual approach) and a technological point of view. The chapter ends with some final remarks about our contribution to the standardization effort in the W3C and some open research lines and challenges for future reference.

1 Multimodal Systems

Nowadays, novel ways of device services and input / output management systems for indoors networking could emerge thanks to the actual wide range of computing devices available with differing capabilities and computational power.

Multimodal interaction

This kind of systems, named *"multimodal systems"* support classic human-computer interaction like mouse, keyboard, display, as they can also support new ways of interaction like gestures, speech, body monitoring, haptics, eye blinks, glove mounted devices or graspable user interfaces. Thus, generally speaking the term "multimodal" refers to this combined and extended variety of modes of interaction.

In this section we will try to define these systems on the light of these new modes of interaction and new research fields opened by the last ten years of changes in the technical realm and in the information technology market.

First of all, we will define the term "multimodal" more accurately.

Related research areas

The term "multimodal" has been used in several disciplines: in Communication Studies [Collier 2004] [Knapp et al. 2009], Psychology [Spence et al. 2004] in Software Design and Human Computer Interaction [Nigay 1994] [Martin 1995] [Duce 2000] [Oviatt 2003] [Bernsen 1994] [Bernsen et al. 2010]. These are some different points of view and approaches that we must address.

Outline

For this reason, first we will define some terms; secondly we will discuss an intended frame of reference for a "multimodal" system. Finally, we will present some systems through the lenses of this frame of reference in order to understand the scope of this notion in our research and in order to explain what a «multimodal» system is today.

1.1 Multimodal Terminology used in this Research.

Terms

We begin with the introduction of various fundamental concepts that will be referred to in the sections below. They will be explained from a human-centered approach and in reference with the multimodal literature. These concepts are *sensor system, effector system, mode, modality, medium, fusion* and *fission*.

1.1.1 Sensor System

Human perception is a process in which we acquire, interpret, select and organize the sensory information with a *Sensor System*. In Psychology, a *Sensor System* is a part of the nervous system responsible for processing sensory information. In this context, a *Sensor System* consists of sensory receptors (sensors), neural pathways, and parts of the brain.

In human perception, a *physical carrier* [Bernsen et al. 2010] is a type of physical phenomenon that can be captured by one of the main human senses: vision, hearing, touch, taste and smell. Examples of physical carriers are light, temperature, chemical impact, sound or pressure.

A *Sensor System* [**Figure 1.1.1**] is an artificial system developed in order to imitate human perception to enhance the human-computer communication [Cariani 1992]. In result, today's computers are capable of receiving sensory information from *physical carriers* as gamma rays, infrared, ultrasound or magnetic fields, going far beyond the human perception limits.

A sensor system is more than a sensor device

As a technological object, a sensor is a device which receives a *physical carrier* input signal and transmits it to a *Sensor System*. For us, cameras, haptic devices, microphones, biometric devices, keyboards, mouse and writing tablets; are devices that can act as sensors and provide sensor services for an artificial *Sensor System*.

1.1.2 Effector System

According to Norman's action cycle [Norman 1988] human action is a process in which we start determining what to do; then we determine how to do it and finally we do it with an actual physical action. This physical action is executed through human effectors that are usually the hands, the voice, or the facial muscles. [Latta et al. 1994]

Seemingly, an artificial *Effector System* [**Figure 1.1.1**] converts information coming from a computer to some *physical carrier* perceptible by humans and executes a physical action in a human's *sensor system*. This is done trough artificial effectors.

An effector system can use effector devices and sensor devices

As a technological object, an effector is an output device that provides sensory stimulation with a *physical carrier*, from an *Effector System* to the user. Some effector devices are : displays, speakers, pressure applicators, projectors, vibrating devices. Even some sensor devices (galvanic skins, switches, motion platforms) can provide effector services for an artificial *Effector System*.

1.1.3 Mode

In current language, a *mode* is a particular type or "form" of something [Nigay et al. 1996] In computer systems the *mode* is determined by the device type selected by the end user or by the system.

Yet, in a *Sensor System*, the captured raw data is processed in a way that gives a particular sensorial *mode* to the captured information. For example, data can be captured in a Visual *mode* [**Figure 1.1.1**].

In *Effector Systems* the *mode* is the particular "form" taken by the returned information. This result is returned in the effector selected by the system or the end user in a way that gives an specific *mode* to the information. For example, data can be returned in an Acoustic *mode* [**Figure 1.1.1**].

The mode is the sensorial form of the data

Therefore for us, a *mode* is a "way" of representing, encoding or decoding the information according to a specific perception kind: a Visual *mode* (everything visible), an Acoustic *mode* (everything audible), a Haptics *mode* (everything tangible), an Olfactive *mode* (everything olfactory) and a Gustative *mode* (everything gustatory).

Adopted definition of mode compared to other researches

This definition corresponds to the "communication mode" in [Bellik et al. 1992], to the Niels Bernsen's "medium" concept [Bernsen 1994], to the "sensory modality" concept in Psychology [Spence et al. 2004] or to the "modality" concept in Martin [Martin 1995] which is a computer process of content synthesis/analysis according to the type of input/output data.

1.1.4 Modality

The modality is the performative form of the data

Modality is a fuzzy concept. It covers the way an idea could be communicated or the manner an action could be performed [Nigay et al. 1993]. In the majority of systems for example, the primary *modality* is speech and an additional *modality* can be typically gesture, gaze, sketch or any combination thereof. These are forms of representing information in a known and recognizable logical structure. For example, acoustic data can be expressed as a musical sound *modality* (a human singing) or as a speech *modality* (a human talking). They correspond to the primary information code in the communication process.

The same mode can have multiple modalities

Every *mode* has at least one modality. On the input side, when a sensor e.g., a microphone captures a sound in an acoustic *mode*. Then an abstraction layer into the *Sensor System* can define its modality, in this case, the speech or the musical sound *modality* [**Figure 1.1.1**]. The semantic information about the *modality* can be given by the *Sensor System* itself or by an external component of *modality* recognition.

On the output side, when a vibrating device is used for guiding in an *Effector System*, the returned raw data can be given in a tactile *mode* by a "tap-on-the-shoulder" vibration *modality,* representing the direction of motion [Van Erp et al. 2005]. In this way a precise *modality* is instantiated according to the semantics of the message to communicate and according to the adequacy between the final *mode* and the communication goals.

Modality processing can be embedded on devices

In both cases, the *modality* management demands the participation of an abstraction layer that can be part of the *Sensor System / Effector System* itself [**Figure 1.1.1**] or delegated to a dedicated component in the system.[4] This choice will depend on the system's architecture.

In conclusion, a *mode* have a limited number of *modalities* that are semantically related to the nature of the *mode* itself and also semantically related to the communication goals.

From this point of view, the notion of *modality* is very dependent on the type of message to communicate and his technical support : his *medium*.

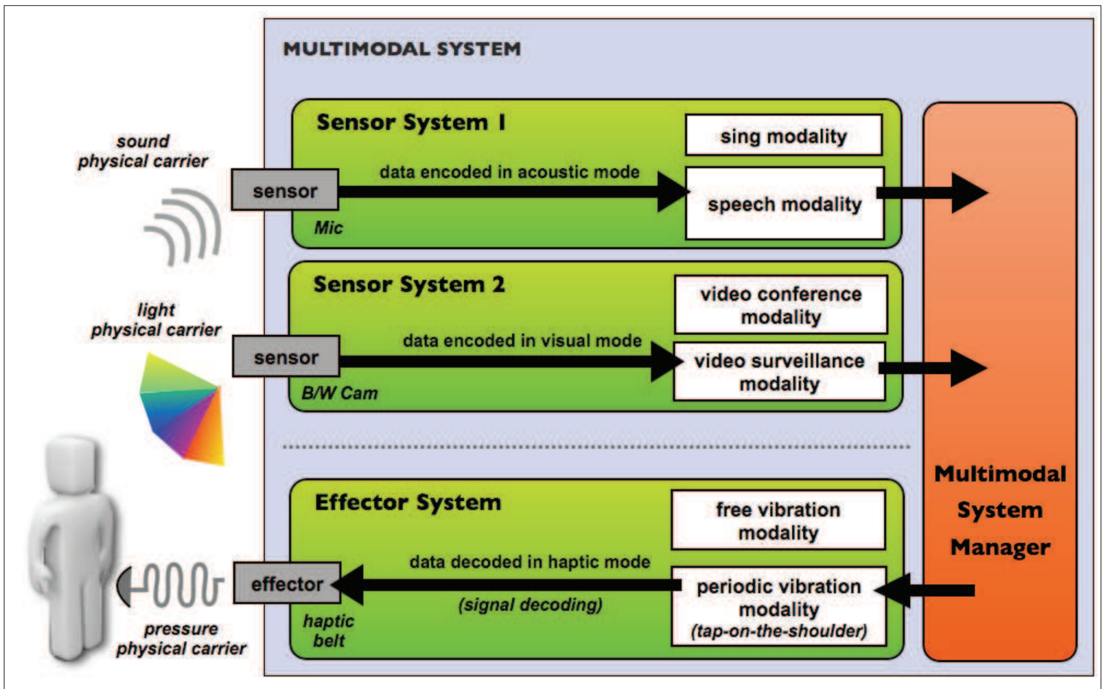

MULTIMODAL SYSTEM
sound physical carrier
Sensor System 1
sensor
data encoded in acoustic mode
Mic
sing modality
speech modality
light physical carrier
Sensor System 2
sensor
data encoded in visual mode
B/W Cam
video conference modality
video surveillance modality
Effector System
effector
data decoded in haptic mode
(signal decoding)
haptic belt
free vibration modality
periodic vibration modality (tap-on-the-shoulder)
pressure physical carrier
Multimodal System Manager

Figure 1.1.1: Sensors, Effectors, Mode and Modalities in a Multimodal System.

1.1.5 Medium

Most researchers in human computer interaction agree that a medium is a technical support for the information [Nigay et al. 1996]. It can be seen as a physical device (e.g., a Tv set, an e-book reader), as a logical entity (e.g., a software, a graph, an animation, an SMS) or as a combination of both.

Adopted definition of medium

We adhere to the third case : a *medium* is a technical entity that is a support for a limited kind of logical entities[5] according to the combined semantics of the message and the capabilities of the support itself. It correspond to the secondary information code in the communication process.

For example, on one hand, if we use a Tv set as a display for a computer, we are not using the television *medium* anymore; because the technical device without the logical content is not sufficient to identify the *medium*.

4 It depends on the policies of device discovery in the multimodal system and the capabilities of the device.

5 See e.g. the limited number of Modality Instantiation Models presented in [Bellik 2006].

On the other hand, if we see a Tv show in a movie theater, this is not the television *medium* neither, because we can not say that we are still "watching Tv". This illustrates that the television *medium* is a phenomenon that relies on the relationship between the semantics of the message with a specific kind of support and a specific kind of use. [Bellik et al. 1992]

The definition takes into account the social dimension of media

This relationship between the technical entity and the logical entity is semantically attached and **socially determined**. The television *medium* is defined by a certain kind of animated images, (produced in a certain socially accepted way) and a technical support associated to this *medium* (e.g. a Tv set) and finally, a precise interaction type. It is a representational system coupled with a precise technical object.

Difficulties of the multimodal composition

One of the difficulties for the production of a multimodal experience through a multimodal system, is that it could be necessary to dissociate the *medium* as a document (a logical entity) of the *medium* as a technical support, in order to compose communicational equivalent documents over different devices.

Another difficulty is that these equivalent documents and interaction mutations must also be user friendly and adapted to the situation of use. When a *modality* change occurs, the interaction type also changes. A multimodal experience imposes In this way a new kind of interaction: an interaction that mutates over time as far the *modality* changes.

We have already seen that in a *Sensor System* an acoustic *mode* can be performed in a speech *modality*. This *modality* can be interpreted as a dictation *medium*, a voice command *medium* or a conversational *medium* [**Figure 1.1.2**].

The medium is a source of semantic information about the user

At this level the "intention" of the user influences the communication process as the social conventions associated to the information "sensed" in an activity or social event. As a result, in a *multimodal system,* the modal recognition may pass through the lens of the semantic context given by the *medium,* in order to capture or instantiate the unimodal or multimodal message.

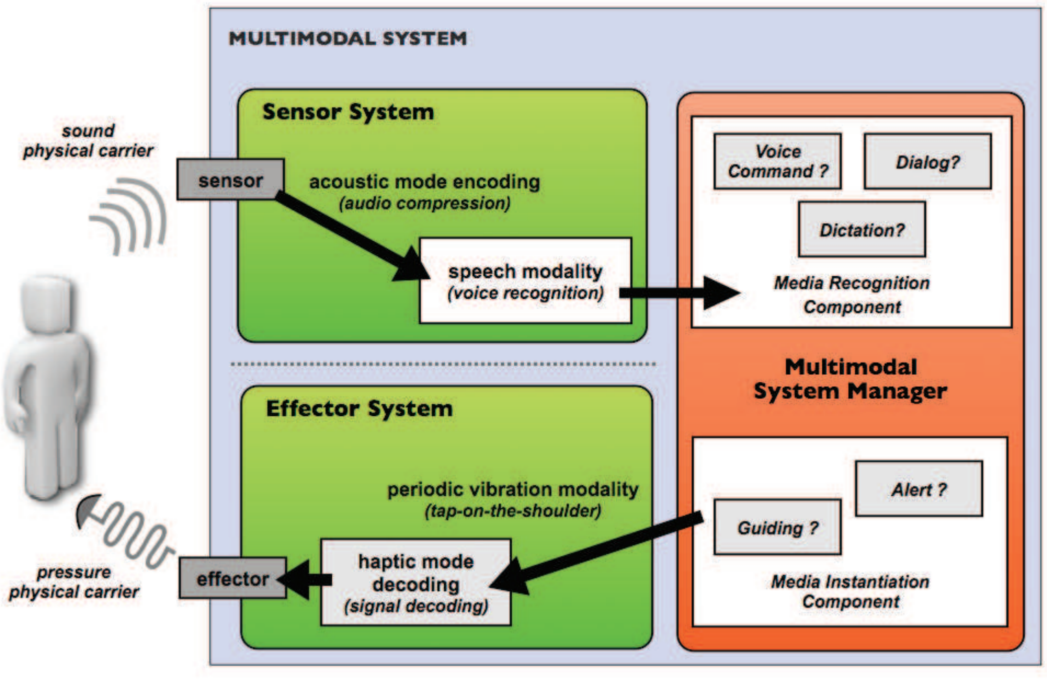


Figure 1.1.2: Media handling in a Multimodal System.

1.1.6 Fusion

Following [Hall et al. 1997] the *fusion* is: *"The integration of information from multiple sources to produce specific and comprehensive unified data about an entity"*. It is also called "multimodal signal integration". The goal of the *fusion* is to integrate the data coming from a set of sensor systems or to integrate data in the actions to be executed by a set of effector systems. In both cases the result is an emergent semantics given by the combined multimodal human-system interaction.

Fusion is a kind of integration

Depending on the system's implementation, the combined result can be integrated at the data level (*data fusion*), at the feature level (*feature fusion*), at the decision level (*semantic fusion*) [Dasarathy 1997] or at the prediction level (*hybrid fusion*) [D'Ulizia 2009].

Adopted fusion types

1.1.6.1 Data Fusion

In the *data fusion* the raw data from multiple similar sensors is combined. The fused result provides a more detailed and more reliable information about the same physical phenomenon and over the same *mode*.

Used to enhance information detail

For example, the *fusion* of multiple temporal sensor video images is used for resolution and contrast enhancement, in the resulting video image [**Figure 1.1.3**]. In the output side, the video overlay of an image in an *Effector System* could be a case of *data fusion*.

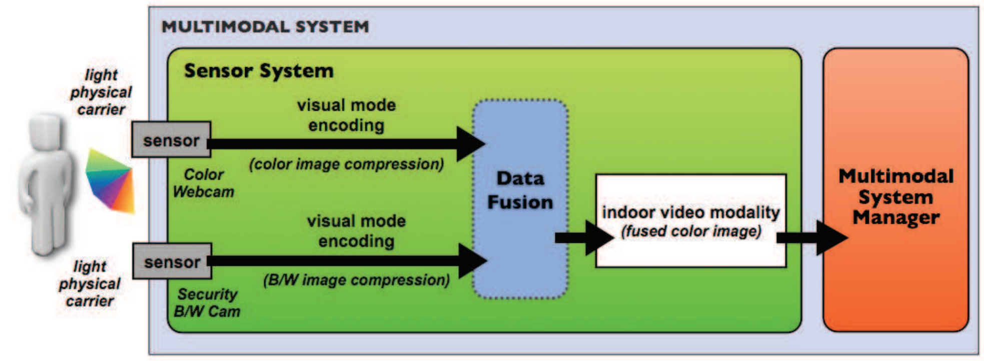


Figure 1.1.3: Input Data Fusion in a Sensor System capturing in visual mode.

1.1.6.2 Feature Fusion

In the intermediate level, the *modality* processing and recognition of one *mode* influences the processing and recognition of the other *mode*. Historically, researchers have applied *feature fusion* in speech and lip movement in which both sensor systems provide corresponding information about the same words and articulated sounds.

Used to synchronize modes

In other cases, in a scene reconstruction *modality*, e.g., the *feature fusion* operation can be made with a network of webcams capturing the same scene from different viewpoints used in conjunction with an acoustic sensor capturing background audio in order to detect the crowd's affective behavior modality [Zhihong et al. 2009]. This way, in the *modality* recognition component the two *modes* (visual and acoustic) are closely synchronized [**Figure 1.1.4**].

The same process can be executed on the output side, giving a synchronized restitution to an *Effector System*, e.g., in a Tv set.

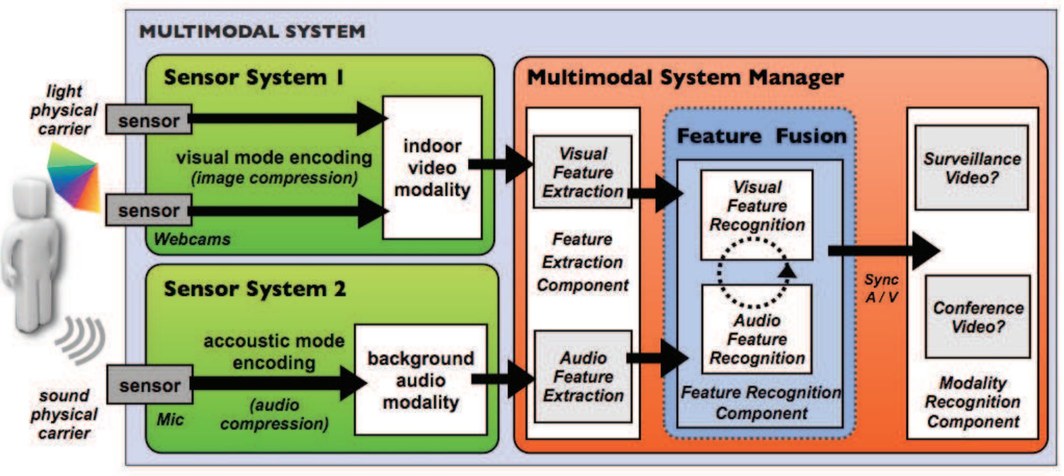


Figure 1.1.4: Input Feature Fusion in a Multimodal System.

1.1.6.3 Semantic Fusion

Used to build a joint interpretation of partial and non synchronized data

In the decision level, *semantic fusion* integrates meaning to manage loosely-coupled modalities using mutual disambiguation techniques [Caschera et al. 2010]. This requires a common semantic representation for all modalities and a well designed process to integrate partial semantics. For example, the semantic information is extracted from each recognized input / output and then merged.

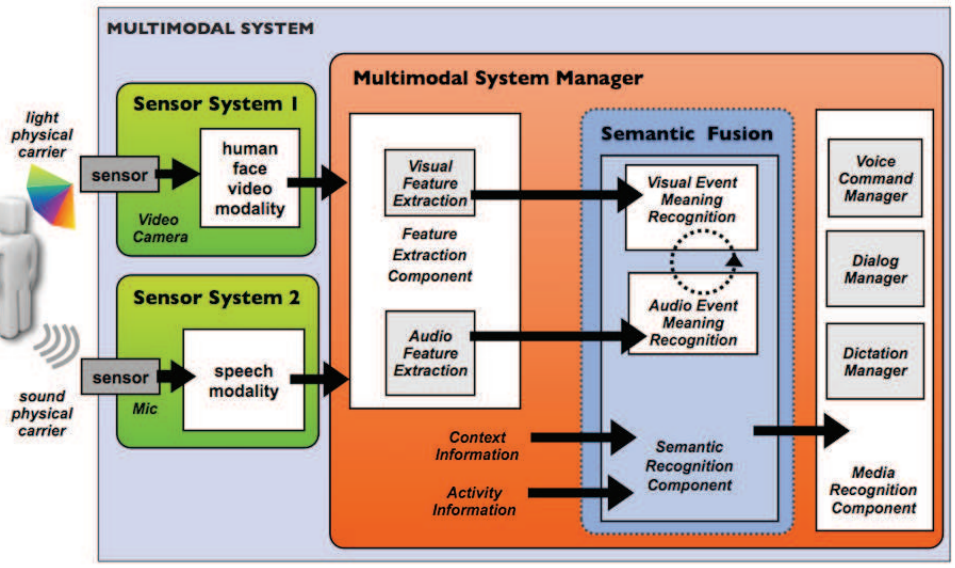


Figure 1.1.5: Input Semantic Fusion in a Multimodal System.

This way, the semantic outcome is separately interpreted before the merging process of the meaning. [**Figure 1.1.5**]

In most cases, the *semantic fusion* is used for modalities that differ temporally. For example, in the combined speech and touch input, when a user says: "professor", and at the same time she draws -sketch modality- a rhombus (< >) containing the label "professor" handwritten. These two modalities are executed with a temporal delay, and fusion is used to detect that both are referring to the same command (redundancy). Then, *semantic fusion* involves merging the semantic content obtained from multiple Sensor / Effector Systems to build a joint interpretation.

This interpretation contains consistent information about the interaction activity with modalities, like current event, localization, physical and emotional state, and so on. Every activity is meaningful only in this particular multifaceted context.

Semantic fusion relies also on the quality of previous processing. During the semantic fusion, the active modalities are matched by grouping the modalities events to obtain a low-level interpretation. After this comparison, the interpretation result is transferred to a high-level interpretation module, to obtain the meaning of these events. This high-level interpretation defines the type of actions triggered by the user, the parameters used and his context [**Figure** 1.1.5], identifies the meaning of the user's behaviors, and finds the most proper association with the user intention in an specific activity or situation. On the output side, the *semantic fusion* is achieved by the adaptation of the functional data of the *multimodal system* to the interaction conceptual needs. The interface design and conceptualization can force some *semantic fusion* of the functional data to present it in an engaging and meaningful way [Coutaz et al. 1991].

1.1.6.4 Hybrid Fusion

In the prediction level, with the *hybrid fusion* the interpretation is distributed among multiple kinds of *fusion* : it can use the data, feature and/or decision levels.

The interpretation process is based on predictions related to the dependencies between previous and current modalities.

The extraction and interpretation process in the data or in the feature level is determined by the semantics previously extracted and interpreted at any of the other levels (the decision level, e.g.). [**Figure** 1.1.6]

A recursive process between all levels of fusion

By considering current information with respect to previous information, a statistical prediction and its probability can be derived from a combined interpretation. Thus, a *hybrid fusion* covers all the available levels of *fusion*. For example, in the use-case presented in Appendix 6-STEP 7 at T1 the user turns his face saying «news», the system fuses image and sound (*data fusion*), then analyses and recognizes the gesture and the sound features and fuses them (*feature fusion*) in a integrated UI update command with the «news» data and finally the meaning of the data is recognized and semantically matched to the application functions linked to this command «news zapping». (*semantic fusion)*

At T2, the *data fusion* can focus on the face recognition given that the latest feature recognition was a face turn, the *feature fusion* can look for similarities in the current command and predict that the gesture feature is probably a zapping command while the sound feature can be a different one, based on the information provided by the semantic level and finally the *semantic fusion* can predict that the zapping semantics are preserved if the current feature fusion in T2 is similar to the T1 value.

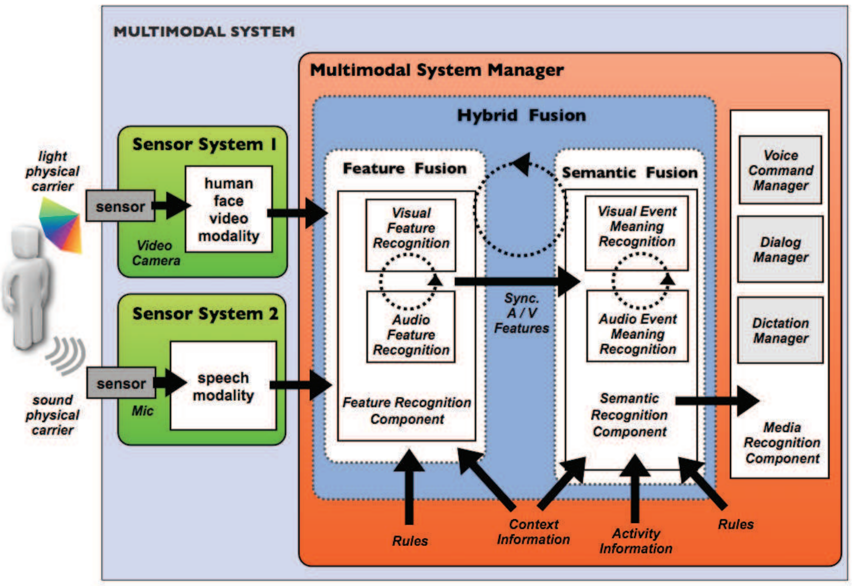


Figure 1.1.6: Input Hybrid Fusion in a Multimodal System.

1.1.7 Fission

Fission definition adopted

In a *multimodal system*, the *fission* refers to the decomposition phenomenon: it is the process of realizing an abstract message on some combination of the available modalities in more than one *mode*. It is also called «response planning», «presentation planning» or «scene composition».

The goal is to generate an adequate message, according to the context of use (in the car, home, conference room), current activity (course, conference, brainstorm) or preferences and profile (chair of the conference, blind user, elderly). It is closely related to the information structure and meaning.

The tasks of a fission component

Therefore, a *fission* component must determine which are the most relevant modalities, select which media is the best content to return with the effector systems available in the given conditions; and coordinate this final result [Foster 2002].

Multimodal *fission* is linked to the repartition of information among several communication modalities in multiple modes. It also resolves which part of the content will be generated within each *modality* when the global multimodal content has been determined.

Adopted fission types

This process occurs before the processes dedicated to the information rendering or restitution in the *Effector System*. As we will detail below, it can be performed at the signal level (*data fission*), selection level (*modality fission*) or at the dissociation level (*semantic fission*).

1.1.7.1 Data Fission

At the signal level, the raw data is sent to the right *Effector System* considering its mode and available modalities. This is typically the case for a video: the sound track being sent to an acoustic *Effector System* and the visual track to a visual *Effector System* [**Figure** 1.1.7] that can be implemented on different devices.

In data fission signals are split

In the process of *data fission* takes place the transformation of content depending of the devices. Fused signal streams may be split into their respective modes and are mapped to the effector systems according to the modalities available in the effector device.

The *data fission* includes also matching the data rate, compression type and format to the capabilities of the effector device and the construction of a coherent and synchronized result: when multiple output modalities are used, layout and temporal coordination are to be taken into account (output coordination) [Dumas et al. 2009]. On the input side, the separation of a visual and acoustic track from a recorded video in a *Sensor System* is also a case of *data fission*.

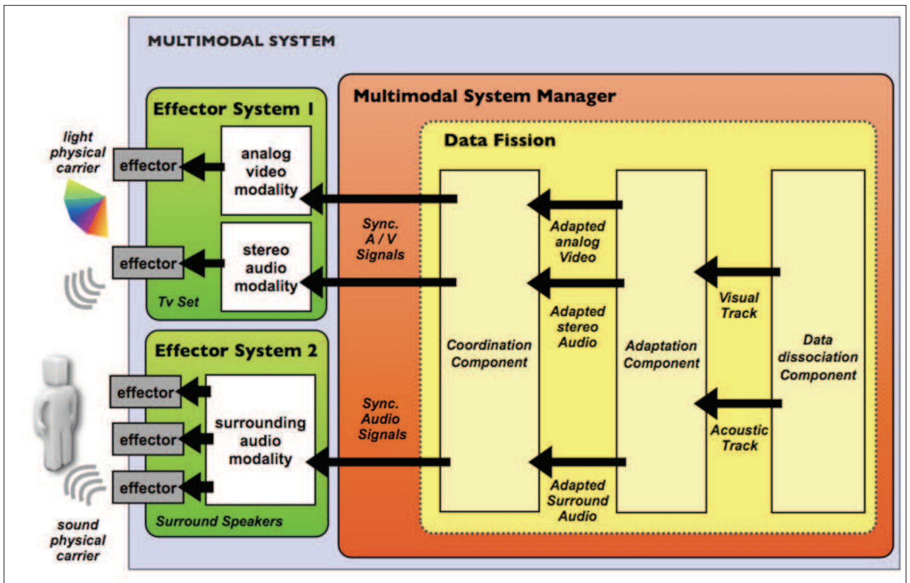


Figure 1.1.7: Output Data Fission in a Multimodal System.

1.1.7.2 Modality Fission

The *modality fission* is the constraint-based repartition of data over the available modalities. The constraints and the information content are given by a semantic component of higher level of abstraction. The goal of the *modality fission* is for example, to affect modalities to elementary information units [Rousseau et al. 2004]. For every unit of semantical information, the *modality fission* operation gives a list of the possible and meaningful modalities [**Figure 1.1.8**].

Another example can be founded on the use-case presented in Appendix 6-STEP3Bis1. Given the unavailability of the sound recognizer modality, the system will affect another available modality to the same elementary information unit, in our case, the search need. Then it composes a new user interface in the HTML web page fusing the new modality (a search text input form) with the already present ones, and deleting the icons corresponding to the speech recognition.

This is a selection process based on rules and semantic descriptions which once applied, add or removes certain modes, modalities or information content candidates, according to the running state of the interaction context and the semantic constraints of the elementary information units.

Used to select and compose modalities

Characteristics such as available modalities, information content description, modality characteristics and task to be performed are forms of knowledge used for the *modality fission* selection. The modalities can be selected also according to context, activity information, social information and user profile in order to convey all data effectively in this given situation.

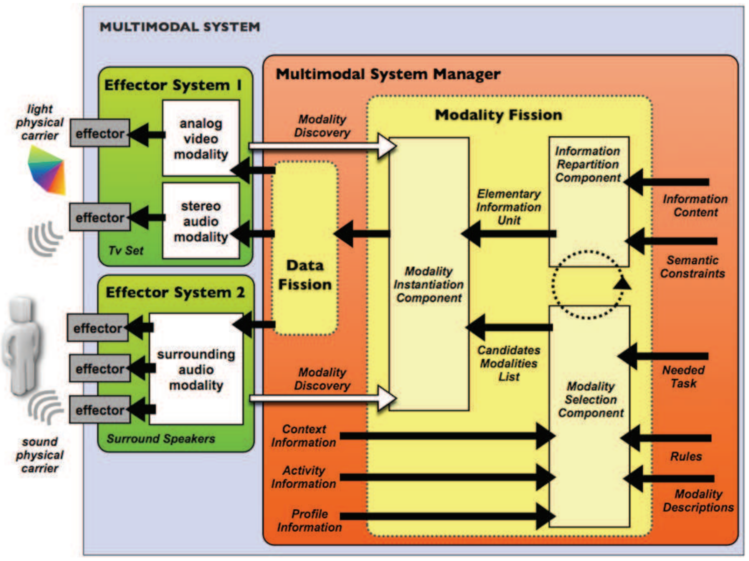


Figure 1.1.8: Output Modality Fission in a Multimodal System.

1.1.7.3 Semantic Fission

The *semantic fission* operation performs the construction of the message, where the information to be transmitted to the user is created according to the communication needs. [Rousseau et al. 2006] In the *semantic fission* the content to be returned must be selected and arranged into an overall structure and the more meaningful modes are evaluated according to the combined and integrated semantic contributions of the different available modalities. This means that the current state of the system can affect the composition of the message. In order to keep the result meaningful, the final structure can be combined according to the description of the target *medium*. For example, the *semantic fission* can return the decision of displaying visually the part of the information that requires persistent attention, and of verbalizing in an acoustic mode the part whose only aim is to capture selective attention.

In the use-case presented in Appendix 6-STEP14 the final presentation is adapted with a *semantic fission* deciding on the need of sound output in the current modalities. The broadcasted sound will have a priority while the synthesized sound in the interface will be deactivated. Finally *semantic fission* can be executed on the input side, mostly in the analysis of speech commands describing a combination of desired outputs : «turn down volume on TV adds» describes the combined behavior of two output modalities (the HTML web page and the image recognizer) controlled by the user input.

Used to handle the context and to compose a multimodal message

In other cases, the *semantic fission* operation may decide to instantiate redundant representations of the same concept over multiple modes [**Figure 1.1.9**]. Both are semantic decisions related to the communication goals, the context, the engaged activity and the user profile.

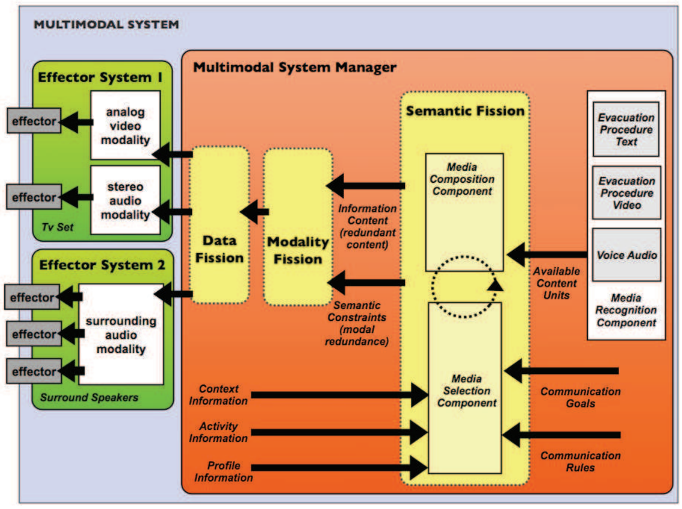


Figure 1.1.9: Output Semantic Fission in a Multimodal System.

In conclusion, we can classify a *multimodal system* through the lenses these terms definitions [**Table 1.1.1**] . The direction expresses the input or output focus, the level of abstraction corresponds to the point of view used to handle the inputs or outputs and the other characteristics will follow the definitions proposed above.

Direction	Input	
	Output	

Level of Abstraction	Input Abstraction	System	*(e.g. Sensor System)*
		Device	*(e.g. input device)*
	Output Abstraction	System	*(e.g. Effector System)*
		Device	*(e.g. output device)*

Supported Modes	Acoustic	
	Visual	
	Haptic	
	Olfactive	
	Gustative	

Multiple Modalities for each Supported Mode ?	Yes	
	No	
Media Support	Yes	
	No	

Fusion	Data	Input	
		Output	
	Feature	Input	
		Output	
	Semantic	Input	
		Output	
	Hybrid	Input	
		Output	

Fission	Data	Input	
		Output	
	Modality	Input	
		Output	
	Semantic	Input	
		Output	

Table 1.1.1: First elements for an analysis of Multimodal Systems.

With these basic terms explained, we can study in the next section a more empirical description of what a *multimodal system* could be today, based on the analysis of existing tools and the mechanisms currently used to address the multimodal challenge.

1.2 Definition of a Multimodal System.

Based on the previously defined terms[6] and on [Maes et al. 2003], a *multimodal system* is for our research[7]:

- A system with multiple *Sensor Systems* capturing raw sensory data from multiple *physical carriers* and coding this data in multiple *modes* structured on the form of one or multiple *modalities*.
- A system with multiple *Effector Systems* decoding data in multiple *modes* in the form of one or multiple *modalities* and returning sensory stimulation in multiple *physical carriers*.
- A bi-directional system with combined inputs and outputs in multiple *modes* and *modalities*.
- A system in which input and output data can be fused or dissociated in order to identify the meaning of the user's behavior or in order to compose a meaningful returning message using multiple *medias, modes* and *modalities*.

Besides these characteristics concerning the multimodal system structure (***what*** a *multimodal system* is)[8], a *multimodal system* designed today can **potentially** encompass other characteristics [Maes et al. 2003] that we will describe in the following sections . We group them into three categories constructed according to the conceptual model proposed in this thesis[9]:

- Characteristics related to the system's and user's behavior regarding the session management, the event handling, the dialog strategy or the decision making process.
- Characteristics related to the description of some interaction participants : devices, users and domain data.
- Characteristics related to the general delivery context of the system: the usage situation, the temporal situation and the spatial situation.

1.2.1 The Multimodal System Behavior.

In the following section we will cover ***how*** a current multimodal system could operate for the management of his behavior regarding a multimodal session, the multimodal events raised by the user or by the system, the multimodal communication acts, and its synchronization and interpretation using some AI decision mechanisms.

1.2.1.1 The Multimodal Session.

A *multimodal session* represents a time interval during which the multimodal application's resources are associated to a user or a group of users and they remains available. It is very important for distributed applications that involves multiple devices and users, because it represents the resources that can be used by an user or an interaction manager in a given time for a given interaction. The multimodal session is a semi-permanent information that represents the addresses, description and states of the available modalities connected to the multimodal system, even if these modalities are not currently used in the interaction.

Over a *multimodal session* a user or a group of users can interact with one or more sensors / effectors and they must be able to suspend and resume the interaction cycle. During the *multimodal session*, the system must also allow them to change of sensors / effectors and with this, to switch to another *modality*. In others terms, during a *multimodal session* the user must be able to interact through several modes and their modalities.

The multimodal session ca be paused and resumed with another modality

The *multimodal session* represents the lifetime of an interaction cycle; it represents the relation between the user identification, the resources and modalities allocated to the interaction and a time interval. It can be ended or paused and the resources can change, but the user or users identified must remain the same. A *multimodal session* can for example, be related to the lifetime of modalities, their availability or their interval of communication with the system.

6 See supra part 1.1. Multimodal Terminology used in this Research.

7 This basic definition corresponds to the foundational documents provided by the W3C and the notions addressed by the Multimodal Interaction Working Group group.

8 This is the (*What*) part of a description of a distributed multimodal system characteristics according to the knowledge model proposed in this document on section 2.2.3.2 A Knowledge Model for Multimodal Discovery.

9 See infra part 2.1.1.2. Abstract Layers of the Architecture and section 2.2.3.2 A Knowledge Model for Multimodal Discovery.

In addition during the *multimodal session*, the interaction can be recorded, because it can be helpful to have a record of the received entries, the returned output, the resources used, the interaction participants, the current data model or the sequence of data model changes made during the various cycles of interaction.

In this way, a *multimodal session* could be used to replicate a state across devices, across processes within the same device and across material services. In result, the use of a *multimodal session* could provide a mean to synchronize multiple modes and their modalities.

A *multimodal system* could:

- Use a multimodal session.
- Handle the session migration between modalities.
- Historize the session.
- Support session management.

1.2.1.2 The Multimodal Event

According to [Bunt et al. 2005] an *Event* is a temporal structure that represents a *communicative act* or a state.

A *communicative act* is an utterance that we use to perform some sort of action in communication [Austin 1962] [Searle 1969]. Basically, it is any instance on human or human-computer communication.

There are three categories of communicative acts : assertive (action to inform about something), imperative (action to command something) and interrogative (action to demand something).

According to the *Event* definition above, we can transpose these categories to multimodal systems, and affirm that some events must be useful to inform things to the *multimodal system* (*notification*), others to command things through the multimodal system (*command event*) and finally some events can be used to ask things to the *multimodal system* (*control event*).

Besides that, events can also be classed [Maes et al. 2003] as asynchronous or synchronous; generated remotely or locally; managed remotely or locally and finally, user generated or system generated.

A definition of the multimodal event

According to [Bellik et al. 1992] in multimodal systems, an *Event* is the basic information generated by the software interface and associated to a material resource (a device). *"A monomodal event is generated by only one device (a mouse click, a word, a gesture, etc.). A multimodal event is a set of monomodal events produced closely in time by different devices."*

In this research we will use the terms *Event* and *monomodal event* indiscriminately, knowing that in any case, an *Event* is characterized by its source, its target, its date, its duration, its type, and its value. This characterization assumes that an *Event* is discrete, time stamped, typed, structured and that it can be of three types: a Notification, a Command or a Monitoring Event. From this perspective, a *multimodal system* could :

- Manage *monomodal* or *multimodal* events.
- Preserve the temporal ordering of events.
- Associate an *Event* with information about the interaction or the delivery context.
- Manage the *Event* granularity. [Bellik et al. 1992]
- Support the *Event* handling, generation, synchronization and disambiguation (in the case of contradictions between events [Fitzgerald et al. 2003]).

1.2.1.3 Interaction Management Strategy

From the mid-1960's with the program "Eliza" [Weizenbaum 1966]; the interaction had being viewed as a dialogue between two partners: a user and a machine. This notion is often referred as the *conversation metaphor*:

"In a system built on the conversation metaphor, the interface is a language medium in which the user and system have a conversation about an assumed, but not explicitly represented world. In this case, the interface is an implied intermediary between the user and the world about which things are said." [Hutchins et al. 1985]

Since the late 1980's the *conversation metaphor* is applied to the graphical user interfaces [Brennan 1990] and then extended to the multimodal interfaces in which, from the beginning with the «Put-that-there» voice and gesture interface [Bolt 1980] the approach was mainly language centric: very often, language is the main modality, and other modalities are used to complete the textual (oral) message.

Nevertheless, some current multimodal systems use also the extended notion of *dialogue acts*[10] which can be performed by linguistic or non-linguistic means and most of them describe the interaction manager as a dialog controller.

The dialog controller is a kind of interaction manager

[Maes et al. 2003] also affirm that technically, a dialog is an interaction between the user and a multimodal application. A dialog involves turn taking, focus handling, intent detection and information disambiguation. For them, dialogs may be classed as *directed dialog*, in which one party follows a predefined path independently to the other parties responses or as *free flow dialog*, in which both parties can control the dialog flow or content at any time.

From a user point of view, when multiple sensor or effector systems are present in a multimodal system, inputs/outputs can be handled individually as separated process that are sequentially used during the *communicative act*, or as a combined input/output resulting from a synergic use [Nigay et al. 1996]. In the first case, for example, in a redundant interaction the user points and confirms with a voice command. Input is in this case treated individually as a sequence of separated process with a common goal but the second input is used to validate the reliability of the gesture input. In contrast, an example of the second case is a synergic use of a microphone and video analysis to recognize speech. In this case redundant input data are combined to produce a more robust recognition.

This corresponds to the *Sequential Multimodality* proposed by [Maes et al. 2003] in which an input or an output is provided by a single modality and can change over the dialog time; or to the *Simultaneous Multimodality* in which an input/output is provided in multiple modalities. The modality composition can also change over time but every input/output is treated separately.

Types of multimodality

In the case of output *Sequential Multimodality* the user can select what modality to use at any time based on his situation and for example, switch from a sequence of video modality in his cell phone to audio output to follow the news while driving. In the second output case, the *Simultaneous Multimodality,* an alert can be provided by a vibration, a sound and a text prompt simultaneously.

However, from a system point of view [Maes et al. 2003] propose a third option: the *Composite Multimodality*, in which input/output may be provided in a simultaneous way but it is handled as a single "composite" input using semantic *fusion* (interpretation) or presented as a single output by semantic *fission* (composition) algorithms to decide what makes sense to combine.

This is the case in the composition of joystick and game commands received on multiple modalities at the same time and treated as a single, integrated compound input by downstream processes.

10 Because dialog acts can be linguistic or non-linguistic, in our research we will use the term *communicative act* to refer to a dialogue act. We consider this term more general and more appropriate to describe multimodal interaction issues.

Multimodal symmetry

These three options reflect also the synchronization behavior that not only covers the way in which inputs are combined but also the way inputs are reflected in the output choice and behavior.

Furthermore, the synchronization behavior can be different depending on the synchronization granularity . This can be made at the event level, the feature level, the medium level or at the session level. [Maes et al. 2003]

Dialog strategy categories

Depending on the system, the interaction management strategies can be different. It has being classified in several non exclusive categories: finite state-based by [Rojas-Barahona et al. 2009] [Bui 2006], frame-based, agent-based by [McTear 2002], information-state, probabilistic by [Traum et al. 2003] or plan-based approaches and cooperative approaches by [Cohen 1997].

In the next section we will briefly describe these categories that could be used by a multimodal system designed nowadays.

Finite state-based	Also named *Dialog grammars,* are very useful in *directed dialog* management. In systems with a well structured task, the dialogue structure can be represented as a state machine transition network in which the nodes represents the predetermined utterances of the system.
Frame-based	Are more flexible. As in a form filling interaction, the information to display can be arranged in an structure gathered according to the semantic relationship between the fields to fill. This structure can be a frame, a schema, a flow / task graph or a type hierarchy. In this way, *frame-based strategies* reflects the dependance of the fields to fill, captures the meaning of the queries and can reflect the information priority for each user.
Information-state	Are based on the distinction of dialogues. They propose to formalize the dialog management in terms of information state updates. With this goal, it is important to identify the relevant aspects of the information in dialogs, how these aspects are updated and how these updates can be controlled. The information state of a dialog represents the cumulative additions from previous actions and the future dialog actions. It is the information that a participant have at a particular point of a dialog: what they brought to the dialog, what they pick up from the dialog and how they are motivated to act in the near future. For example, while statements give information as propositions, questions provide motivation to others to give statements as answers. Thus the information state of a dialog based on questions will have a motivational load in its information state more important than a dialog based on propositions. In this way, a model of dialog states as information states and can be used to manage a *free flow dialog*. The information-state helps to distinguish a dialog from another, because context, mental state or conversational scores can be included in an information-state entity. Finally, the information-state can be represented as a list, a set, a record or a logical inference. It contains the *communication acts* on the form of dialogue moves that can be updated according to some rules and an update strategy.
Probabilistic	Allow dynamical changes on the dialog management based on the optimization of the cost or reward related to the current information-state. By this means, probabilistic strategies enables the dialog system to statistically learn . It is done using Markov Decision Processes, Partially Observable Markov Decision Processes, inductive logic programming, Q-learning or Bayesian Networks techniques.
Plan-based	Consider communication as a goal fulfillment activity. The idea is that utterances are also actions at different levels, and the *communicative act* is used by humans to achieve relational or material goals. The strategy is organized around the discovery of the emitter's underlying plan and the selection of the appropriate response. These strategies are mainly based on the action planning theories originated from [Cohen et al. 1979] and they are useful, e.g. in conversational games design
Cooperative	Also named *Collaborative strategies)*, the *communicative act* serves to transmit and renew knowledge, in a process of achieving mutual understandings. It then coordinates action towards integration and solidarity of the dialog participants as it was proposed for human communication by the Communicative Action Theory [Habermas 1981].

In these strategies dialogue is regarded as a joint activity, something that agents do together. This model claims that both parties to a dialogue are responsible for sustaining it. To collaborate in a dialogue requires the participants to have at least a joint commitment to understand one another, and these commitments motivate the clarifications and confirmations so frequent in ordinary conversation.

This joint activity can be useful in agent-based approaches for problem-solving issues, for meaning and multimodal negotiation and for achieving goals by collaborative sharing of knowledge, beliefs and intentions.

Considering all these interaction management issues and communication models, a current *multimodal system* might :

- Support sequential, simultaneous or composite multimodality.
- Support seamless synchronization behavior at some level and manage the synchronization across modalities.
- Handle which communicative partner (e.g. the user / users or the application) is the next to interact during a multimodal dialogue. This is also known as turn taking management.
- Update the interaction context based on the interpreted inputs.
- Support direct interaction, free flow interaction or both.
- Manage the focus of the multimodal interaction
- Contain and maintain the state of the multimodal application.
- Update the communication exchanges history.
- Support grammars, structures, planning, learning or intention/emotion interpretation.
- Implement one or more dialog strategies.

1.2.1.4 Decision Making

Decision needs in a multimodal system

A *multimodal system* may need a support on decision making in some features like: *perception*, *behavior* management, *knowledge* representation and in *situated* and *social* multimodal intelligence.

a) Perception

In a *multimodal system*, the input *fusion* is the basis of any input "*perception*" and the *fusion* of the inputs given by multiple sensors can be performed thanks to a large number of decisions taken at the *data* level, the *feature* level, the *semantic* level or at the prediction level.[11]

For instance, if we focus on the case of feature level, speech and lip movements are two complementary inputs given by two separate sensor systems.

In *feature fusion* the recognition component in the *multimodal system* must "decide" which feature in one *modality* correspond better to a feature in the other *modality*, in order to compare the recognized sounds with the lips movements and interpret them in a fused result.

11 See supra part 1.1.6 Fusion

The use of different modalities can also result in one or more interpretations of the user input and consequently, a decision must be taken in order to resolve any given ambiguity using, for example, rules or classifiers.[12]

In multimodal researches, this process is called ambiguity management [Caschera et al. 2010] and involves finding the most proper interpretation of the user's behavior captured in each modality and the most proper synchronization of the multiple modalities involved in the current multimodal interaction.

Types of decisions

According to [Dubois et al. 1991] this is a choice problem in which a decision mechanism makes a "suggestion" based on logical knowledge and static constraints. These constraints can be heuristic rules in the form of decision tables of three types: *imperative tables*, *focus tables* or *advice tables*.

A decision table can represent the necessary conditions that should be respected in order to satisfy constraints; this is a *imperative table*. A decision table can also be used to eliminate certain decisions and only keep what is relevant but postpone the resulting choice; this is a *focus table* and finally, a decision table can represent a piece of advice on the way to guide the constraint propagation process toward a solution; this is an *advice table*.

Decision techniques based on strong and weak AI

Several formal and non-formal approaches[13] can be used to choose the technique to exploit any of these decision tables.

The first approach, is an "all or nothing" manner, where the data must correspond exactly to the description of the situation.

A second approach, is dedicated to reasoning under uncertainty, in which the situation descriptions can be represented in terms of fuzzy sets and decision tables (or predefined classes) are exploited by means of patterns matching. This second approach can be also completed by a decision ranking [Dubois et al. 1991] or by a negotiation process that aggregates decision tables preferences (or predefined classes) in an statistical way.

It can be done using Support Vector Machines [Bredin et al. 2007], Bayesian Networks [Town 2007] [Choudhury et al. 2002] [Atrey et al. 2006] [Garg et al. 2003], Dempster-Shafer Theory [Reddy 2007], Hidden Markov Models [Gaitanis et al. 2007] [Nefian et al. 2002], Relational Graphs [Chai et al. 2004] or Neural Networks [Ni et al. 2007].

In a probabilistic way, decision making can also be handled with Kalman Filters [Talantzis et al. 2006] or with Particle Filters [Nickel et al. 2005]. At the end, if the ambiguity is still present and decision is in a deadlock state, it is also possible to apply some weak methods, as *means-end analysis* (MEA)[14] or the *hill climbing* method [15] to reach a final solution. These decision techniques are useful not only for handling the *"perception"* of the multimodal system but also can be used in *behavior* management.

b) Behavior Management

As we showed above, in a *multimodal system*, output *fission* and output *fusion* are the basis of the system's behavior towards the user. As multimodal interaction takes place in highly dynamic and unpredictable conditions, reactive and proactive planning is needed in order to respond to any incoming interaction.

This process requires to make choices that maximize the utility of the available choices and then, based on these choices, setting goals and achieve them during a defined period of time. In this way, decision making is a key pillar for *action planning* in dialog management, *modality fission*, *semantic fission* and any *sensor / effector* coordination.

12 A classifier is a function that uses a pattern matching technique to determine the closest match.

13 The research in decision issues for multimodal systems can be classified in two kinds: formal or symbolic approaches inspired by the classical AI research (named Good Old Fashion AI - GOFAI [Haugeland 1985]) and non-formal approaches inspired by the so-called "weak" nouvelle AI. Synthetic, embodied, situated or behavioral AI are some examples of this new approach.

14 Given a current state and a goal state, a chosen action must reduce the difference between the two. The action is performed on the current state to produce a new state, and the process is recursively applied to this new state and the goal state in a goal-seeking form.

15 It is an algorithm that starts with an arbitrary solution to a problem and then, it attempts to find a better solution by changing in an incremental way a single element of the solution. If the change produces a better solution, an incremental change is made to the new solution, repeating until no further improvements can be found: this result is the final iterative solution.

c) Knowledge Representation

Furthermore, in a *multimodal system,* the *knowledge* representation will be influenced by the features supported by the decision making mechanism chosen for the *feature fusion*, the *semantic fusion*, the *hybrid fusion*, the *modality fission* and the *semantic fission*.[16]

The system needs to "understand" internal[17] and external states, time, goals, entities (devices, services, modalities, media), participants (users or systems), properties, categories and relations between objects. To comprehend which is its current self-knowledge, its domain knowledge and its general knowledge, the system needs to make choices and decision techniques are there to support these choices.

Finally *situated* and *social* multimodal intelligence are important features that also need a concrete decision making management. A dominant and recurrent argument coming from the *situated* intelligence researchers [Cassel 2001] [Wooldridge 2002] is their conviction that human knowledge will never be attained by modeling it as something static, rigorous and inflexible when it is dependent on body phenomena, dynamic, evolving and nuanced.

d) Situated Intelligence

Thus, *Situated* intelligence is based on the premise that intelligent behavior is not disembodied[18], but it is a product of the interaction between an artificial intelligent system and its environment.

In multimodal interaction most of the phenomena depend on the surrounding background, and to reduce it into a number of factors that could be manipulated by a machine can not drive to a fully understanding of the real-world situations. [Bringsjord et al. 2007]

This could explain the increasing presence of social sciences and cultural studies in the multimodal interaction field, who questioned the validity of classical laboratory experiments. They call for a greater focus on ethology[19] or comparative psychology; and on real interaction exhibited in the real world (as opposed to the highly constrained conditions of the laboratory).

Decision based on situated intelligence

Moreover, *situated* intelligence has already shown that symbolic representations of the world are simply not necessary for solving a wide variety of problems and that many problems can be more efficiently tackled by doing away with representations and exploiting the structure of the surrounding environment [Brooks et al. 1998].

Situated intelligence in multimodal systems works heavily with computer vision and learning [Bohus et al. 2010].

For example, in [Arsenio 2004] autonomous agent perception and multimodal object and events recognition uses one or many of the decision techniques described above, even if the behavior of the agents is not guided by a high-level representation of the world to support their decisions.

This *situated* decisions can also depend of the *social* intelligence[20] needed to understand the multimodal application background. According to [Dreyfus 1992] "*...understanding requires giving the computer a background of common sense that adult human beings have by virtue of having bodies, interacting skillfully with the material world, and being trained into a culture*".

16 See supra part 1.1. Multimodal Terminology used in this Research.

17 The understanding of an internal state covers not only the functional state but also the emotional state of the system. In other words, for us, *emotional* intelligence is part of the general knowledge representation of the system, as a reflexive knowledge of it-self.

18 *Situated* intelligence is also known as *Embodied* Intelligence.

19 Ethology is a combination of laboratory and field science, with a strong relation to other disciplines such as neuroanatomy, ecology, and evolution. Ethologists are typically interested in a behavioral process. Rather than in a particular group of individuals they often study one type of behavior in a number of unrelated individuals.

20 *Social* intelligence in humans was originally defined as "the ability to understand and manage men and women, boys and girls, and to act wisely in human relations" [Thorndike 1920]

Following this idea, *situated* perception is a kind of data preparation in which humans "predigest" a knowledge that is deeply dependent of what human life implies, for example, being *"trained into a culture through social practices of human society (involving being born by parents, going through childhood and adolescence and growing up and learning personal responsibility, social interaction, making friends, and establishing an identity...)"* [Larsson 2005]

e) Social Intelligence

Thus, *situated* intelligence is deeply related to *social* intelligence even in a pre-symbolic stage, which is clearly showed in recent robotics multimodal systems trained by humans as learning children [Brooks et al. 1998] [Arsenio 2004] or in embodied conversational agents decrypting social gestures like Rea [Cassell 2001].

Social intelligence definition

For computer systems, *social* intelligence is defined as "the ability of an agent to build a social relationship with others and to use it when solving a variety of problems, and the ability of a group to learn from experiences when solving problems." [Nishida 2010]

Social intelligence researches in multimodal systems includes the analysis of a) non verbal social interactions in smart environments [Eagle et al. 2006] [Ohmoto et al. 2010] and robotics [Mohammad et al. 2009], b) the studies on situated Knowledge Management [Merckel et al. 2009], c) on Social Signal Processing [Pentland 2007], d) on Implicit Interaction [O'Grady et al. 2011], e) on Reality Mining [Greene 2008], and f) on Role & Dominance Studies [Hung et al. 2007].

In a general way, all these *social* intelligence researches take also advantage of the decision mechanisms described above.

To summarize, decision techniques are involved actively on the functioning and structure of many current multimodal systems. And this regardless of the system's complexity or the perspective about the kind of knowledge needed for its operation. In a general way we can say that concerning the decision making mechanisms, a *multimodal system* designed today could:

- Support choices to satisfy constraints, eliminate inadequate options or recommend options.
- Help in any interpretation or ambiguity management process.
- Help to reasoning under uncertainty in highly dynamic conditions.
- Implement one or more formal or non-formal decision techniques.
- Collaborate in reactive and proactive planning of the interaction.
- Collaborate in the understanding of real-world conditions and analysis.
- Participate in the construction of self-knowledge, domain knowledge and general knowledge, useful for the interaction and for the final users and groups involved.

In this section we have covered some possibles characteristics of a *multimodal system* regarding the management of the multimodal session, the multimodal event generation, the interaction's synchronization and its interpretation using strategies coming from the Artificial Intelligence field.

In a multimodal distributed system, the session management can be strengthen by the use not only of *notification* and *command events* but also with the use of control events responsible of the handling of state information. Finally, this information will enhance the integration and composition of modalities performed during the phase of synchronization on any of the strategies described above. As we can see, these three aspects of the behavior of a system become then, complementary.

Thus, we covered in this section *How* a distributed multimodal system could behave and some possible aspects (this list is not extensive) to take into account in a Reference Architecture. This is the first part of our description of *Which*[21] kind of multimodal systems can need such an architecture.

Depending on the implementation and the application needs, a *multimodal system* can have one or many of these attributes and in that case, we can classify *multimodal systems* following these behavior factors [**Table 1.2.1**] :

21 This is the second part (*How*) of a description of a distributed multimodal system characteristics according to the knowledge model proposed in this document on section 2.2.3.2 A Knowledge Model for Multimodal Discovery.

Session	Supported	
	Handle by Manager	
	Migrated	
	Historized	

Event	Supported		Temporal Order	
	Handle by Manager		Monomodal Support	
	Synchronized		Multimodal Support	
	Disambiguated		Associated to context	
	Multiple Granularity			

Interaction (Dialog)	Multimodality	Sequential		Interaction Types	Direct	
		Simultaneous			Free flow	
		Composite			Both	
	Synchronized			Focus Handled		
	Historized			State recorded		
	Turn Taking			Context updated		
	Strategies	One		Interpretation	by grammar	
					by structure	
		Multiple			by planning	
					by learning	
		None			by intention int.	

Decision	Goal	Satisfy constraints		Techniques	Formal	
		Reduce options			Non Formal	
		Recommend			None	
	Knowledge	System		Uses	Resolve ambiguity	
					Uncertain reasoning	
		Domain			Support interpretation	
					Collaborate in planning	
		General			Support RW understanding	

Table 1.2.1: System Behavior Elements for an analysis of Multimodal Systems.

1.2.2 Participants in a Multimodal System

In order to launch and support any communicative act, a multimodal system needs to have some knowledge about ***with whom?*** it interacts. A participant can be a system component or some external entity.

Both must be previously described to the system in order to effectively handle their behaviors according to their communication capabilities. These participants can be devices, some kind of users and some specific domain data.

In the following section we will introduce the issue of the participants description and the most common description techniques of devices, users and domain under the light of the description requirements that a *multimodal system* could need.

1.2.2.1 Device Modeling

Devices can be present directly or through share mechanisms implemented in the near or extended network surrounding the *multimodal system*. In both cases, in order to use one or multiple devices in a *sensor system* or in an *effector system*[22]; or in order to have a knowledge about the current state of the participants, the *multimodal system* need to have access to the description of the device capabilities.

The device description form

This description is a static component that provides to the access mechanism the information about the device characteristics including device's form, manufacturer name, version, serial number, system resources, available modalities, control commands or his level of synchronization.

22 See supra part 1.1. Multimodal Terminology used in this Research.

In this case the metadata description is often represented as a document, a *device profile* that is implemented with technologies like CC/PP[23], Delivery Context Ontology [W3C-DCO 2007], FIPA Device Ontology [FIPA-DeviceOnt 2002], MPEG-21[24] [ISO-MPEG21 2002], the Electronic Device Description Language [IEC-EDDL 2006], CANOpen EDS [CENELEC-CANOpen 2002], the Transducer EDS [IEEE-1451 1993], GSDML [IEC-GSDML 2003], SensorML [OGC 2000], FDCML [IDA-FDCML 2001], a simple property/value list or an XML document.

This information can be accessed through a driver interface (e.g. a TWAIN Dynamic Link Library), a device API (e.g. the DirectSound API in a Windows OS, the OSG-i Device Kit, the API Bridge on a Symbian phone, a flash.sensors API in a Flash Player or any of the Device API's of a Browser[25]) a device configuration file, a general station description (GSD) file or a device description repository component like the Device Description Repository [W3C-DDR 2008], WURFL [WURFL 2011] or the MODBUS Device Directory [MODBUS 1979].

However, the device description can be also represented as metadata delivered by a service embedded on the device. This is the case in UPnP networking, where after the bootstrapping process, the control point can retrieve the device's XML description in an URL provided by the device in the discovery message.

This is also the case in DPWS [26] [OASIS-DPWS 2009] in which the same description mechanism is fully aligned with the Web Services architecture, using XML Schema [W3C-XMLSchema 2011], WSDL [W3C-WSDL 2001] and WS-Policy [W3C-WSPolicy 2006] recommendations.

During the discovery process, the device's abstract model used by the *multimodal system* is crucial, due to this variety of device description kinds and their access mechanism.

Unfortunately, this descriptions can be very heterogeneous, because the number of devices available in the market is currently exploding. Furthermore, the granularity of the description of the device features determines the size and utility of the *device profile* itself. A *device model* for a *multimodal system* must give an useful categorization of features to store a pertinent information in a *device profile* with a data quantity and a data format that are still manageable in the current conditions of multimodal systems.

Categories of devices

In a *multimodal system*, devices can be categorized as either unimodal devices or multimodal devices. Unlike other categorizations like controlling devices / controlled devices, sensor devices / effector devices, input devices / output devices, in which a given device may embodied both roles, a categorization based on the number of modes supported by the device has only two options. However, this implies also that a great number of contemporary devices are, in fact, multimodal devices.

A *device model* for a *multimodal system* may take in account these categorizations in order to describe the technical object in a generic and useful way. For example, taxonomies can address physical properties (such as motion and pressure), the data that a device handles (discrete or continuous) and the dimensions of device's inputs.[27] [Buxton 1986] [Card et al. 1991]

Current descriptions of devices

Devices may contain logical devices, as well as functional units, or services. For example, a functional categorization of devices is currently defined by the UPnP protocol with 59 standardized device templates[28] in which one type, the Basic Device profile is a generic template. In the same spirit, the Echonet consortium [Echonet 2002] defines 6 Device Groups for 61 Classes of Devices. In both cases, the device specification defines explicitly the device's properties and access methods.

23 The Composite Capability Preference Profiles Specification of the W3C. [W3C-CC/PP 2010] or even with the The User Agent Profile Specification extension of CC/PP maintained by the Open Mobile Alliance (formerly the WAP Forum). [OMA-UAProf 2001]

24 This is made in the Specification's Part 7 : Digital Item Adaptation, in which Terminals and Terminals capabilities are described.

25 For example any of the ConfigurationData attributes created according with the Media Capture API Specification of the W3C. [W3C-MCAPI 2010]

26 Devices Profile for Web Services

27 See Infra. Section 2.2.3.1 A Glossary for Multimodal Discovery.

28 Over 5000 different types that exist worldwide, according to [Dibowski et al. 2011]

If we analyze the UPnP architecture [UPnP-Arch 2008] two general classifications of devices are defined: controlled devices and control points. A controlled device acts as a server, providing its description and in some cases, responding to remote calls from control points that can send control messages to the control URL of the service (provided in the device description).

Control messages are expressed in XML using the Simple Object Access Protocol [W3C-SOAP 2003]. Then the service returns action-specific values in response to the control message. The effects of the action are modeled by changes in the variables that describe the run-time state of the service.

Hence, in UPnP for the control point to interact with the device, it must retrieve the device's description from the URL provided by the device in a discovery message. When a control point is added to the network, the UPnP discovery protocol allows this control point to search for devices of interest on the network. The fundamental exchange in both cases is a discovery message containing a few, essential specifics about the device and one of its services, e.g., its type, universally unique identifier, and a pointer to more detailed information.

In DPWS [OASIS-DPWS 2009], the candidate successor for UPnP 1.0, the selection target is a device and the hosted services do not participate in the discovery process due to network limitations. The selection mechanism specifies the type of the device or a "scope" in which the device resides, or both.

Device descriptions based on action intent

The "scope" notion is a set of attributes than can be used to categorize devices in a logical and more semantic way. Scope metadata can be used to organize services into logical groups [OASIS-WSDiscovery 2009], (e.g. a given space in a building) or they can be classed by Actions supporting an specific kind of "intent".[29] In this way, the device modeling takes in account a more generic description than UPnP descriptions.

Considering the device description and modeling issues, a multimodal system could :

- Have a mechanism to request device description information.
- Have a mechanism to keep this information available to other participants.
- Have a device's abstract model in order to handle this kind of participants.
- Use a generic categorization of devices and features with multimodal pertinent information.
- Support device discovery and selection based on a multimodal device model.
- Take on account semantic metadata about the use of participant devices.

1.2.2.2 User Modeling

Nowadays the personalization process in computer systems is intended -among others things- to help people with information and services overload. When relevant information about the people interacting with a system is stored explicitly rather than embedded within the logic of every application or service, a *user model* is needed [Kobsa 1989].

Adopted user model definition

The information about the user that is relevant for the system is organized in an abstract data structure (metadata) known as the *user model*. It is an abstract representation of the user that computers can understand and exploit. It can be handled by the system itself or delegated to a specific external profiling system.

29 As defined in http://developer.android.com/reference/android/content/Intent.html for an Android development an intent is «an abstract description of an operation to be performed». For further information see [Kinlan 2010] An Intent is an "intention" to do an action. It is basically a message to say you did or want something to happen. Depending on the intent, a target android application or the OS might be listening for it and will react accordingly. To listen for an intent like the phone ringing, or an SMS is received, a broadcast receiver must be implemented. To fire off an intent to do something, like pop up the dialer, an intent must fire off saying which action will be triggered. The available intents are given by the android platform API.

Approaches to implement a user model

In the former case, three approaches can be used to define a *user model* in a system [Carmagnola 2011] : a top down approach with a predefined abstraction of the user; an induction-based approach that allows making general assertions about the user based on relevant data collection; and finally a mixed approach in which deductions based on an abstract model are made during the phase of "cold-start"[30] but afterwards these deductions are confronted with collected data inductions.

In the case of a *multimodal system* this means that multimodal input and output processes can serve as data collectors but also that the multimodal input and output instantiation can be guided by an abstract and predefined *user model,* based on theoretical assumptions about the multimodal user. This model can be as simple as a preferences categories or complex as stereotypes categories based on collected data. In any case, the model will guide the construction of a list, a document or even an ontology of information to be filled by automatic mechanisms, by the designer or by the user itself.

To sum up, the multimodal *fusion* and *fission* processes will need, a) a basic *user model* in order to guide the selection decisions of *mode, modality* and semantic interpretation; and b) *fusion* and *fission* can help the collection of user interaction data to enrich the construction of a *user model* by induction.

A user model for multimodal interaction

Unlike other systems, the interaction mode in a *multimodal system* is crucial, because the challenge is the personalization of the perceptual experience produced by the exchange with the machine with multiple sensorial modes. This challenge for multimodal user modeling can be viewed at three levels: the sensory level, the functional level and the semantic level.

- At the **sensory level**, sensor systems can capture or be based on physiological features (fingerprint, eye color, scars, body size, etc...) and effector systems can affect or be adapted to this features [ISO-MPEGV 2011] [OASIS-XCBF 2011] [DoJ-JXDM 2005].

This is the first and fine-grained level needed by a multimodal *user model*. Here, the *user model* can produce a *user profile* in a very low sensorial level and can also be the basis to provide multimodal services to another kind of systems. For example, the multimodal *user profile* can be used as a complementary mechanism of verification for applications implementing the OpenID authentication [Lynch 2011] [Sovis 2010].

In order to handle the sensory level of granularity the *multimodal system* must have a *user model* that allows sensory and physiological data collection in a multimodal *user profile*. In a more current case, a preferences file can cover the description of the user at physical level.

- At the **functional level**, the *modality* selection can capture information about behavioral patterns (gait, signature dynamics, habits, etc...) and can also direct the behavior or the perception of the user in a predefined way [ISO-MPEGV 2011] [Pisanelli et al. 2003] [Vainio et al. 2008].

This is a more abstract level of granularity that can be useful for a multimodal *user model*. In this case, the *user model* provides a *user profile* containing assumptions based on the interpretation of behavioral patterns over time. These assumptions will be represented by a preferences list associated to a specific kind of users. This can be also the basis to provide multimodal usage services for systems with very dynamic interaction modes and multiple personalization profiles as for example, in applications with nomadic or distance learning services.

To manage the functional level of granularity, the *multimodal system* must have a generation mechanism of the *user profile*. This mechanism, for example, can link the *fusion* and *fission* processes to a decision component that uses the collected behavioral information to return user description patterns. The resulting knowledge about the user can be stored in a multimodal *user model* if it is capable to support attributes with *situated* intelligent data.

30 This is the phase of the user model initialization, when the user model does not store enough data and little knowledge about the user is available.

- At the **semantic level**, the *media* selection and the interaction activity enriches the information about preferences and social patterns (preferences, demographic data, friends, life-style, etc...) and refines the content and *modality* selection according to the dynamic position of the user in respect to predefined social groups [Heckmann et al. 2005] [Abel et al. 2010] [Reinecke et al. 2011] [Liu et al. 2006] [WP2-FIDIS 2005].

This is the more generic level of granularity in a multimodal *user model*. In this case, the *user model* provides a *user profile* containing deductions based on user models derived from social sciences.[31] These models can be oversimplified classifiers on the form of heuristic archetypes/stereotypes [Rich 1979] [Kobsa 1993] [Kay 1994] that can be used for user classification and social behavior analysis [Niu et al. 2010] [Esposito et al. 2009] [Ghazarian et al. 2010] [Cassell 2009] [Kuflik et al. 2009]. This can be the basis to provide multimodal information services for data, social and reality mining purposes.

To manage the semantic level of granularity, the multimodal system must have a deduction mechanism of the *user profile*. This mechanism, for example, can link the *fusion* and *fission* processes to a decision component that handles the stereotypes and archetypes of the *user model* and use them as classifiers to return a statistical or a probabilistic *user profile* including assumptions about non verbal and implicit social intelligent data. One example of this technique is the current mechanism of user profiling based on social graph analysis and geolocation mining in the web 2.0 and the social web applications.

Where to put the user profile ?

Whatever approach is used to produce a *user profile*, there will still be the question about the access to the profile data from the system modules, since it can be managed in a centralized way, a distributed way or a mixed way.

In the first case -the centralized way- [**Figure 1.2.1**], a user modeling component of the multimodal system (**A**) or a user modeling external server (**B**), can integrate all the knowledge about the user in an unique base.

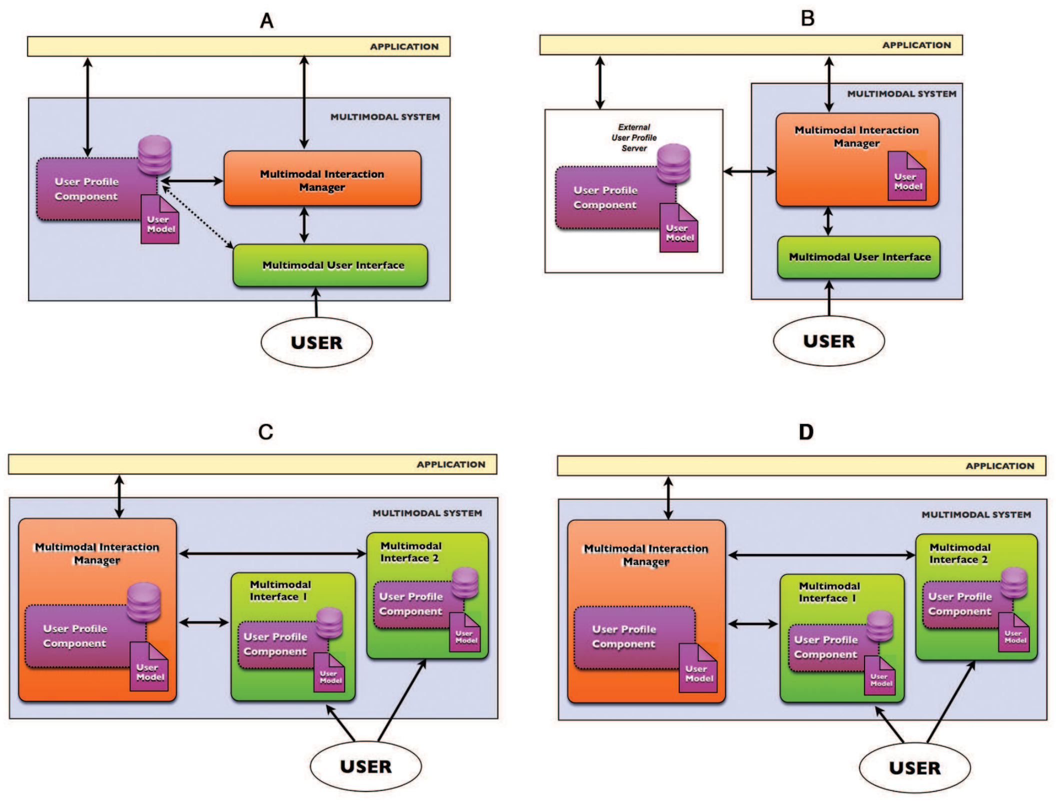


Figure 1.2.1: User Profile Management in Multimodal Systems.

31 For example the affective user models derived from psychology and cognitive sciences [Ortony et al. 1988] or the preferences user models used in social networks and marketing analytics.

The single user model is shared and enriched by the interacting components and the centralized knowledge about the user is shared and made available for multiple components of the system or multiple multimodal applications.

These are generic user modeling solutions that can suffer of the problem of central point of failure, or in the case of several components or servers, the problem of data synchronization and coordination increasing the cost. Finally, in the case of an external server, the system has to cope with the privacy and data protection issues given by an unique storage point.

In the second case -the distributed way-[**Figure 1.2.1**], each part of the *multimodal system* can maintain a small user model as needed for its own purposes (**C**).

In this way the user model is distributed among the *multimodal system*, and it has to combine partial user data from a collection of fragments in a meaningful way.

In this case, each component have his own physical and conceptual representation of the user model and the system has to provide an effective communication layer and exchange mechanisms to enable the share of the user data.

Finally, in a mixed way, the user model can be physically distributed but conceptually centralized. Each component have his own *user model* that refers to a centralized shared *user model* containing the most used concepts (**D**).

If a current *multimodal system* want to address the user modeling, it could :

- Organize the information about the user in an abstract data structure called *user model* that can be imposed, deducted or inferred.
- Have a mechanism to collect user data with its components.
- Support output personalization based on an *user model*.
- Produce or use a *user profile* in a sensorial and physiological level.
- Take on account or capture information about behavioral patterns.
- Support a *user profile* containing deductions based on oversimplified social classifiers and a social-oriented *user model*.
- Store the physical and conceptual *user model* in a centralized, distributed or mixed way.

1.2.2.3 Domain Modeling

A *domain model* is a suite of coordinated abstractions for a formal description of the operating constraints of computer systems in a particular application or domain space. In a *multimodal system* the main concern of the domain is the relationship between the application data and the user interface management in multiple modes and with multiple modalities.

The domain constraints

We can recognize four specific kinds of domain operational constraints : application-dependent operations, presentation operations, interaction operations and usage operations.

These four constraints can be formally described by the means of well-established models helping to produce reusable architectures, reusable components, and a great deal of automated code generation. Each model can have a static part completed by a dynamic part. In this way, a multimodal system may have a *domain model* covering an *application model*, a *modalities model*, an *interaction model* and a *use model*.

- The *application model* is responsible for **providing** concepts of the application business, information about the application situation, and application rules. It describes the application states that reflects the business situation relevant to the user interface. It can describe **which** information the system offers. To model multimodal applications it is necessary to consider the life-cycle of an application [W3C-MMI 2011].

This life-cycle can be divided into four parts: design-time, load-time, run-time and sleep-time. At design time the application model guides the creation, maintenance and enhancement of the application proposing features e.g., with API's or description documents.

The application life-cycle

At load time, the system composes, adapts and loads the application following the application model information, rules or any kind of heuristics. At run-time the end-user invokes a particular instance of the application model and interacts with it's services on a particular hardware device. At sleep-time the application enters on stand-by mode and must keep «alive» the most important interaction data and the application status information.

- The *modalities model* describes the interaction objects (modalities) of the multimodal user interface. It can be viewed as a kind of presentation model, oriented to multimodal systems. It describes the multi-sensorial presentation components of the user interface which are relevant to engage or stimulate the interaction.

To model the modalities it is possible to use theoretical tools given by taxonomies like [Foley 1984] [Buxton 1986] [Mackinlay et al. 1990] [Jacob et al. 1992] for device modalities or [Bernsen 1993] for representational modalities and media.

Tools to model the modalities

- An *interaction model* can be a task-based model which supports the application logic and the interaction flow (dialog). It can define the meaning of the input / output actions; of the valid sequences of actions, and the sense of actions [Bellik et al. 1995] [W3C-IndieUI 2011]. It can describe **what** means every particular action or sequence of actions in a system-oriented way. It can describe **what** the actions **represent** and how the actions are executed.

To model the multimodal interaction, it is possible to use theoretical tools given by multimodal frameworks like TYCOON[Martin et al. 2001] or by multimodal design spaces like the Multi-Sensori-Moteur space MSM [Nigay et al. 1993] or the CARE properties [Coutaz et al. 1995].

Tools to model the interaction

For example, the MSM model analyses the multimodal interaction properties over six dimensions that must be taken into account by designers in the conception of multimodal systems[32].

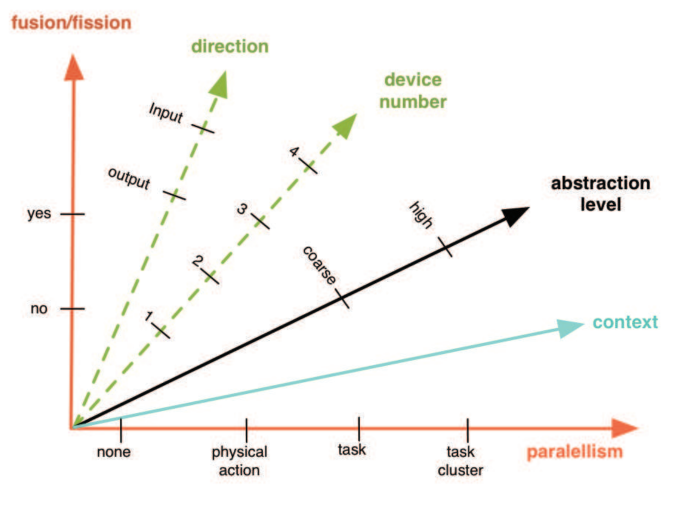


Figure 1.2.2: The MSM space.

These dimensions are :

- Direction: Input or Output.

- Device Number: This is the number of functions of capture or functions of restitution associated to the devices .

32 This is the author's definition of a design space: a theoretical tool to support and conceptualize the design choices.

- Level of abstraction: Represents the level of transformations to apply to the captured or returned information. Two levels: coarse or high.

Context: A set of state variables used by internal process to control the information capture or restitution.

- Fusion/Fission : The combination of multiple units of information to form a new unit or the inverse process, the explosion on multiple units of information from one information unit.

- Parallelism: This is the perception in the interface of simultaneous granularities like the physical action, the task and the cluster of tasks.

The MSM model is a theoretical tool for a designer who can decide which axis to cover and how. It can be used also as an evaluation tool.

In addition to this model, the CARE properties define various forms that multimodal interaction can take and were proposed as a simple way of characterizing and assessing aspects of multimodal interaction that can occur in a given time interval between the modalities available and from the perspective of notions like state, goal, modality and temporal relationship. They are:

- Complementarity: All the modalities must be used to reach a target state and none of them taken individually can cover the target state.

- Assignment: This property represents the absence of choice. A modality is assigned to reach a target state and no other modality can be used to reach this target state.

- Redundancy: The modalities have the same expressive power and they are equivalent used within the same interval of time. Multiple modalities can be used to reach a target state and they convey the same meaning.

- Equivalence: This property represents the availability of choice between multiple modalities but does not impose any form of temporal constraint on them (they don't need to be simultaneous, for example). Multiple modalities can reach a target state if it is necessary and sufficient to use any one of the modalities.

While Equivalence and Assignment express the availability but the absence of choice between multiple modalities to perform a given task, Complementarity and Redundancy describe the relationships between modalities and for this reason, require *fusion* mechanisms.

The CARE model can be used as a tool to characterize interaction with multiple features, for usability testing in multimodal systems or as a characterization of system features in multimodal devices, languages and tasks.

Finally the TYCOON model defines six strategies for multimodal interaction that can be used when a user interface is tested with a focus on the cooperation and synergy between the user and the system. The TYCOON typology proposes to view the interaction as an information transfer or information equivalence or information redundancy, an information specialization or an information concurrency.

To achieve a common goal this model proposes six types of primitives for an informational point of view:

- Transfer: Multiple modalities cooperate by transfer, when a chunk of information produced by a modality is used by another modality. For example, when a mouse click provokes the display of an image, or when a play command by voice launches a video.

- Equivalence: When a chunk of information can be processed as an alternative, by either of them. It means alternative modalities and either cognitive or technical differences between each of them must be considered.

- Specification: When a specific kind of information is always processed by the same modality. It can be modality-relative specification (e.g. using always sounds in alerts) or data-relative specification (e.g. errors only uses sound and no graphic neither text modality can be used).

- Redundancy: When several modalities cooperate and the same information is processed by these modalities.

Complementarity: When different chunks of information are processed by each modality but have to be merged by a *fusion* mechanism.

Concurrency: When different chunks of information are processed by several modalities at the same time but must not be merged.

The TYCOON model extends the CARE model with the transfer and concurrency views. Finally the three models can be used in a current multimodal system to design the behavior and rules used by a *fusion* or a *fission* engine based on these theoretical categories of multimodal interaction.

- A *use model* can define the semantics of the input / output actions on the light of the application needs. The *use model* gives the necessary information to the interactive **performance** of actions, e.g., a payment operation defined by the interaction model as a form and a submit button must be performed in a specific and «trustful» way.

By the nature of this interaction the system must **perform** the actions and interpret the user actions in an specific manner because it could be necessary a) to temporize, b) to add other modes, c) to confirm or d) to add some help to the operation in an user-oriented interaction design. It must reflect the user intent and mental state more than the functioning mechanism of the device or the application.

The use constraints described by the *use model* are the semantics of the interaction depending on the application needs. In other words, it describes semantically **how** the actions will be executed by the participants of the interaction. For example, some of these features can be implemented using tools like EmotionML. [W3C-EmotionML 2011]

Tools to model the application use

It also describes user interface objects, relationships and actions in abstract terms. It can be an extension of an abstract presentation model [Pinheiro 2001] which provides a conceptual description of the structure and behavior of the interface in terms of social rules and emotional intelligence semantics.

If the design of a current *multimodal system* want to use domain models, it could :

- Model the application addressing design-time, load-time, run-time and sleep-time concerns.
- Model the interaction objects using taxonomies.
- Model the interaction with an action granularity or action sequences granularity.
- Model the interaction using classifications, frameworks or design spaces.
- Model the performance of actions based on use semantics.
- Describe user interface objects, relationships and actions in abstract terms that can describe the performance of tasks.

In this section we have covered some characteristics that could be present in a *multimodal system* to facilitate the communication between the participants of the interaction. Depending on the implementation, the system can have one or many of these attributes regarding the devices used for the user-computer interaction, the user profile management and the concerns of the multimodal interaction domain related to the business model. These are three of the most important participants in a multimodal interaction that can be handled by a *multimodal system*.

This section corresponds to the (*Who*) aspect of the description of a current multimodal system[33]. It differs from the classic model < user, device, modality, dialog, domain> because we are addressing the problem not as an IHM issue but as a context-awareness problem focusing on the Situation of use of the system. Based on these participant modeling issues, we can also classify a *multimodal system* following these factors [**Table 1.2.2**] :

33 This is the third part (*Who*) of a description of a distributed multimodal system characteristics according to the knowledge model proposed in this document on section 2.2.3.2 A Knowledge Model for Multimodal Discovery.

Device Modeling	**At Model Level**			**Abstract Description**		
	At Profile Level			**Multimodal Info**		
	Managed	by Document				
		by Api				
		by Service				
		by Repository				
User Modeling	**At Model Level**			**Data collection**		
	At Profile Level			**Personalization**		
	Generation	Deducted		**Levels**	Sensorial	
		Inferred			Behavioral	
					Semantic	
	Stored	Centralized		**Stereotypes used**		
		Distributed		**Social support**		
Domain Modeling	**Application**			**Interaction**		
	Presentation			**Use**		
	Life-cycle	Design		**Granularity**	Action	
		Load			Sequence	
		Run			Mean	
	Interaction Classification	Categories		**Abstract**		
		Frameworks		**Taxonomies used**		
		Design Spaces		**Performance Semantics**		
		Temp. Relations				

Table 1.2.2: System Participant factors for the analysis of Multimodal Systems.

1.2.3 The Interaction Context of a Multimodal System

In this section we will cover some attributes concerning how to manage ***for what?***, ***when?*** and ***where?*** a *multimodal system* can operate. This three issues are related with the general idea of delivery context which is a founding concept in any pervasive or ubiquitous system.

The notion of *context*[34] most cited by authors in pervasive computing is Dey's definition : *"Any information that can be used **to characterize the situation of an entity**, where an entity is a person, place or object that is considered **relevant to the interaction** between a user and an application, including the user and the application themselves"* [Dey 2001]

In Dey's terms, a context implies an **interaction** performed in a given **situation** that we can describe. For him, to describe this situation we must focus on concrete entities as places, persons or objects.

Context as an intentional phenomenon

Yet, an interaction *situation* can be perceived not only as a list of participant entities, but mainly as a relational phenomenon guided by a purpose or finality [Crowley et al. 2002].

In other terms, the context is all the information describing the evolving external factors of the interaction and not only the participants enumeration and description. This dynamic factors are usually triggered by the decisions of the participants in reaction to changes in the the social and spatio-temporal environment. This means that intentionality factors have also to be considered to describe or explain a particular context.

Context modeling approaches

Among the major classifications of context modeling approaches we can find the key-value modeling, graphical modeling, object oriented modeling, logic based modeling, markup scheme modeling and ontology modeling [Strang et al. 2004] [Ejigu 2007].

34 The word is derived from the Latin *con-* (with) and *-texere* (to weave) that describes context as «an active process dealing with the way humans weave their experience within their whole environment, to give it meaning» [Bolchini et al. 2007]

Key-value modeling is the simplest category of models but it is not very efficient for sophisticated and structuring purposes and it supports only exact matching and no inheritance. In contrast, graphical models are particularly useful for structuring, but usually they are not used at the instance level. Examples include the use-cases of UML [OMG-UML 2007] and the context modeling language extension of ORM [Halpin 2009].

Object oriented models [Fowler 1997] views the system as a group of interacting objects. In this approach each object represents some context entity of interest and is characterized by its class, its state, and its behavior. This kind of model has a strong encapsulation and reusability feature but requires low-level implementation agreements between applications to ensure interoperability. Thus they are not suited for knowledge sharing in dynamic systems.

Logic-based models uses logical expressions to define conditions on which a concluding expression may be derived from a set of other expressions. In this approach context is defined as facts, expressions and rules with a high degree of formality. [Barwise et al. 1983] [Akman et al. 1996] [McCarthy et al. 1998]

Markup scheme models uses standard markup languages or their extensions to represent context data and are commonly used for profile representation. For example CCML [Kagal et al. 2002] , CSCP [Buchholz et al. 2004], CC/PP [W3C-CC/PP 2010] or CDF [Khriyenko et al. 2005]

Finally, ontology-based models represent context data and its semantics using RDF triples [W3C-RDF 2004]. RDF is based upon the idea of making statements about the terms covering the entities of the context and expressing the relationships between them. These descriptions of the context can be shared between systems, creating In this way a common conceptual ground for every specific domain. For example, for pervasive environments the CONtext ONtology [Wang et al. 2004]; for context management, CoBra-ONT [Chen et al. 2004a]; for the support of agent-based applications, SOUPA [Chen et al. 2004b]; for the collaboration of devices [Christopoulou et al. 2005]; for context aggregation, CoOL [Strang et al. 2003]; to distinguish relevant context information, mySAM [Bucur et al. 2006]; for mobile systems, SWIntO [Oberle et al. 2006]; or for environment profiles, CoDAMoS [Preuveneers et al. 2004].

Our perspective to define a multimodal context notion

In the following sections we will see how a *multimodal system* could handle the *context* and could use one our various of these modeling tools. We will consider the *context* under this perspective: a multimodal context is an interaction *situation* that we can describe as the relationships between the participants under the umbrella of an intentional purpose in a given space-time.

As [Crowley et al. 2002] stresses, in most use cases interacting directly with the system is not the final intention of the user: the system serves mainly as a mediator or as a tool in real world activities with other humans, with human's creations or with the concrete world.

For this reason, according to [Bettini et al. 2009] in complement to the participant modeling, the context model is currently viewed from a high level approach which aims to also model real world situations; e.g., the usage *situation* of a *multimodal system*.

A definition of the situation concept

According to the New Oxford American Dictionary a *situation* is "*a set of circumstances*", it is a collection of moments, locations, activities and social rules that defines an experience. On the other hand, in the early 1980s Jon Barwise and John Perry attempt to provide a theoretical foundation for reasoning about common-sense and real world situations in the context of natural language processing for computer systems.

For them, "*there are parts of the world, clearly recognized (although not precisely individuated) in common sense and human language. These parts of the world are called situations. Events and episodes are situations in time, scenes are visually perceived situations, changes are sequences of situations, and facts are situations enriched (or polluted) by language*." [Barwise et al. 1980]

With this paper they founded the program of research on *situation* semantics which is an unified mathematical theory of meaning and information content, based on intuitions.

Outline

In the following sections we will explore how are these situational models of the real world, and how they could be used in a *multimodal system*.

First we will present the intents to modeling a *situation* from the point of view of the collection of activities and social rules needed to interact with the system: the *usage* situation.

Secondly, we will present the model of a situation from the perspective of a collection of moments: the *temporal* situation modeling.

And finally, we will see how modeling a *situation* for a *multimodal system* can be related to a set of locations: the *spatio-temporal* situation modeling.

1.2.3.1 Usage Situation

Definition of usage

According to the same dictionary, the *usage* is "*the action of using something*" in an habitual way. It is an action performed as a customary practice. As we can see, the *usage* is slightly different than the use. It implies a behavioral pattern that can be identified as a custom, which is "*a traditional and widely accepted way of behaving or doing something that is specific to a particular society, place, or time*". In other words, the *usage situation* is related to some kind of behavioral pattern of an individual or a group that is socially recognized.

Generally, in computer systems these customary practices are modeled at the granularity of the activity, followed by the task and finally the action. For example, in UML, the activity "Get book from bookstore" can be composed of the task "Buy a Book" which have the actions "Find the book" and "Pay the book".

If the activity is mediated by an on-line store, the usage *situation* of the system is the event of shopping. Because the shopping *situation* is a pattern of behavior socially recognized in occidental societies; it is a custom to which a system can propose a limited series of features. For the same reason, in every given *situation* the number of **commonly accepted** user behaviors is restricted according to cultural constraints.

This example shows how the *situation* model provides a description of the events in the real world with a more abstract granularity than the activity-task-action model. In fact, they complement one another.

For example, [Schilit et al. 1994] observed that the context implies more than the user's location, because other facts of interest as for example the user's social situation are also changing: "*People's actions can often be predicted by their situation. There are certain things we do when in the library, kitchen, or office*".

Situation context management approaches

Since the beginning, the context-aware researches have tried to confront the challenge of a context layer derivation of a higher-level or context detection from a model description of real world phenomena. The most common approaches are:

- The research in **Situational Context** for smart objects [Schmidt et al. 2001], [Gellersen et al. 2002]. In linguistics, the Situational Context refers to every non-linguistic factor that affects the meaning of a phrase. Nearly anything can be included in the list, from the time of the day to the people involved in the location of the speaker or the temperature of the room.

Situational context also refers to the aspects of the real world that suggest what activities normally take place here and now. In humans, this type of social knowledge requires an understanding of the social qualities of the environment including the location, the time period, the particular occasion, and the general policies and values of the surrounding culture.

Mental models as tools for context management

Based on previous experiences in a given situational context, people start developing mental models of these situations, just as they build mental models of people. These models allow people to associate particular architectural forms with functions and behaviors, allowing people to more rapidly process the situation.

People have learned to understand the particular meaning of specific situations, thereby realizing that a solemn funeral is an inappropriate place to scream the latest football scores.

In conclusion, for this approach a Situational Context mostly means the *"multi-faceted characterizations of a situation that typically require substantial analysis and fusion of data from individual sensors."* [Gellersen et al. 2002]

- The research on **Situation Models** [Crowley et al. 2002], [Crowley et al. 2006], [Dobson et al. 2006], [Loke 2006], [Mattioli et al. 2007],[Ding et al. 2007], [Henricksen et al. 2006], [McCowan et al. 2005], [Barraquand et al. 2010], [Brdiczka et al. 2010], [Kaenampornpan 2009].To this approach, representing situations has to take into account the structure of the system as comprising sensors at one level and inference procedures to reason with context and situations at the other level.

They also consider how to manipulate situations as first-class entities and how to reason with a representation of situations within a logic programming language.

Situations modeled as first class entities

Some authors [Barraquand et al. 2010] affirm also that situation models can be useful for the construction of software systems and services for observing and understanding human activity and social interactions.

For this approach, a situation model is commonly defined as consisting of entities and the relations between those entities; or a set of relations between entities where entities are sets of properties.

- The research on **Event Models** [Scherp et al. 2010], [Troncy et al. 2009], [Shaw et al. 2009], [Lagoze et al. 2001], [Crofts et al. 2009], [Raimond et al. 2007], [Bennett et al. 2004], [van Hage et al. 2011].

Event Model definition

An event model is a formal representation of events that allows for capturing and representing occurrences in the real world. This representation can be oriented to describe the "factual" aspects of events, characterized in terms of ***what*** happened, ***where*** did it happen, ***when*** did it happen, and ***who*** was involved.

"Factual" relations can be intended to represent a fact without necessarily being associated with a particular perspective or interpretation [Troncy et al. 2009]; or a fact mediated by human interpretations [Scherp et al. 2010].

In this approach, the event definition that is meant to guide the modeling process is far to be consensual.

An event that refers to facts (factual event) can be viewed as:

- something very generic; from something that happens and is subject to news coverage to any physical, social, or mental process, event, or state.

- something perceptible by a human that is any arbitrary classification of a space/time region, by a human or an «intelligent» software agent. For example, the load-time in a web page is a kind of perceptible event.

- something existing in a time interval, for example, a geographic phenomenon like a tsunami or the fall of the roman empire. These are perduring entities or phenomena that unfold over time and during a limited extent of time.

- something that bind other entities. This is a state or event consisting of one or more objects having certain properties or bearing certain relations to each other, for example the pregnancy event that can also mark a transition between situations.

Thus, the models differ in their focus, domain specificity, size and level of formalization. As the others situation models do, they intend to describe both historical events in the broad sense as well as events in the sense of states or activities.

In summary, these three approaches address the challenge of a high-level model of some real world phenomena in terms of situations that can be used to handle some kind of periodic use of services or applications: a *usage situation*.

If a *multimodal system* wants to support the *usage situation* modeling, it could :

- Model customary multimodal practices in an usage situation model or trace them in a usage situation profile.
- Model social qualities of the environment like the particular occasion and the general policies of the environing culture, that could be useful for the management of the multimodal interaction.
- Model commonly accepted mental models[35] of situations of usage.
- Model "factual" aspects of situations of usage or usage events.
- Model the interpretations or "versions" about usage situations or usage events.
- Model relationships between usage situations.

1.2.3.2 Temporal Situation

The interpretation of Time

In linguistics, a temporal relation is a relation between propositions that communicate the simultaneity or the ordering in time of events or states [Longacre 1983]. In any temporal relation the **Time** entity can be considered as a discrete addition of points: «*times corresponding to instantaneous events*» [Allen et al. 1985]; or as continuous interval that is semantically defined: «*times corresponding to events with duration*».

This two approaches may be mixed: a *time interval* can contain a number of *time points* as a discrete linearly ordered domain with a precise starting and end boundaries [**Figure 1.2.2**]. They can also be distributed over time in an homogenous or in an heterogeneous way.

Nowadays, in addition to these two classical approaches, the research on non formal AI and the research on statistical patterns have proposed a third perspective : a **Time** with some duration for which one of the limits is unidentified [**Figure 1.2.3**].

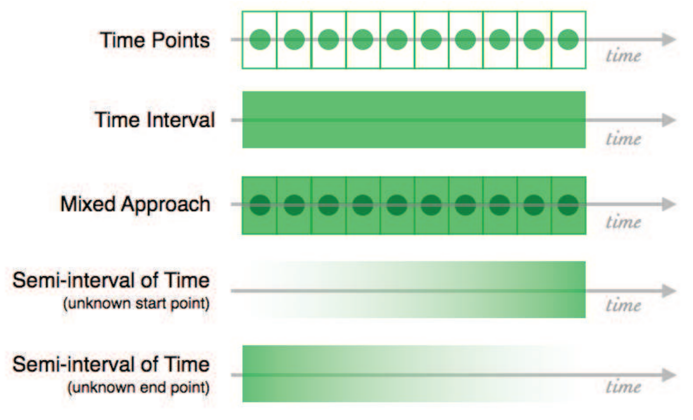


Figure 1.2.3: Temporal Concepts needed in a Multimodal System.

In other words, a continuous interval for which one boundary can be unknown. This is called a *time semi-interval* and it is mostly used to represent an incomplete or a coarse temporal knowledge. [Rainsford et al. 1999] [Möerchen et al. 2010].

35 See: [Johnson-Laird 1983]

Besides, the **Time** can have multiple dimensions that are semantically different. This defines the difference between a series and a sequence.

The Time can have multiple dimensions

On one hand, a *time series* is a set of unique time points [**Figure 1.2.4**], an ordered set of values of a variable while a *time sequence* is a multi set of time points in more than one semantic dimension, a multi-variable set of values that appears on time with some order.

On the other hand, an *interval series* is a set of non overlapping time intervals [**Figure 1.2.3**] while an *interval sequence* can include overlapping and equal time intervals in multiple semantic dimensions. [Möerchen 2007]

In a *multimodal system*, these temporal concepts can be useful for modeling the temporal situation in a large number of tasks: event handling, strategies of *fusion* or *fission* of modalities, turn-taking management, user profile management, etc.. and this, because almost every aspect of an interaction cycle involves **Time**.

Generally two temporal perceptions can be considered in computer systems: the *valid time* and the *transaction time*.

Temporal models in computer systems

The *valid time* is a concept used in temporal databases: this is the time for which a fact is true in the real world. It denotes the time period during which a database fact was, is, or will be valid in the modeled reality, and corresponds to application-time periods. Usually, the time interval is closed at its left bound and open at its right bound. For example, On April 4, 1975 a father registered his son's birth. An official will then insert a new entry to the database stating that John lives from April, 3rd. Notice that although the data was inserted on the 4th, the database states that the information is valid since the 3rd in an attribute Valid-From. The official does not yet know if or when the baby is dead so in the database the attribute Valid-To is filled with infinity (∞). In the case of a multimodal system, it can represent the validity of an interaction cycle distributed over multiple media, modalities and situations.

The *transaction time* denotes the time period during which a fact is stored (and «sensed») by the system generally, the time interval is closed at its left bound and open at its right bound. In our example, it represents April 4, when the real fact was registered, and the date in which the official decide to delete or «archive» the information registered about this son who was born the 3rd April. In a multimodal system it is used for the session management and the registering of the availability of the components.

Both representations of **Time** (*valid time*)capture the information in a discrete way reflecting the **«sensorial»** level.

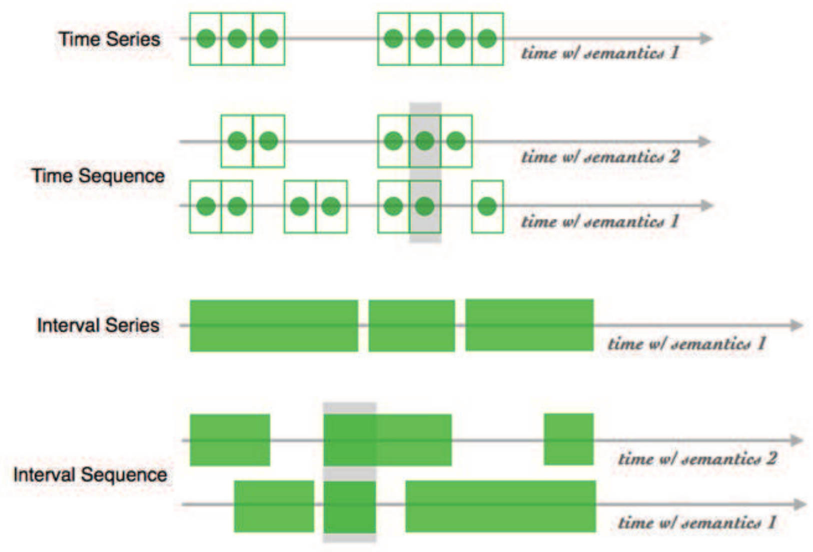


Figure 1.2.4: Temporal Dimensions needed in a Multimodal System.

In addition to these two temporal representations the *symbolic time* is used. This mechanism implements a domain and high level perspective, in which **Time** is represented by symbolic labels. These labels of *symbolic time* can be orthogonal to the *valid time* and the *transaction time*.

For example, in a form a user selects the hour in which he wants to be woken-up: for the system this could be a *valid time* information. The voice interaction to select the wake-up hour is executed at 10 pm : for the system this could be a *transaction time* information. If the user want to be waked-up only in working days: for the system this could be a *symbolic time* information orthogonal to the *valid time*. After configuring the alarm the user has the habit of checking his mailbox: for the system this could be a *symbolic time* information orthogonal to the *transaction time*.

With the symbolic labels, *time points, time intervals* and *time semi-intervals* can be annotated describing their participation in a less formal or measurable informational entities, for example, the participation in social situations or events.

Finally, temporal data models are also affected by the duration of the «perception». Some phenomena are captured in a single long perception (represented by a continuous stream); other phenomena are captured as many short perceptions (represented as a discrete addition of states).

The Temporal Operators

While the notions behind a temporal data model cover the discrete description of phenomena, temporal operators describe the temporal relations between the time pointed data or between the *time intervals* or *time semi-intervals*. They capture the dynamics of the information in a way reflecting the «**functional**» level.

Any operator can take into account one or more of these aspects:

- concerning the phenomenon properties, its *duration* (persistence over time) and its *periodicity* (repetition with a constant pace);

- concerning its relation with other phenomena, the *order* (sequential occurrence), the *synchronicity* (parallel occurrence), the *coincidence* (intersection of occurrences) and the *concurrency* (closeness in time without any particular order). [Möerchen 2006b]

Operators like the algebra proposed with the Allen's 13 temporal interval relations [**Figure** 1.2.5] are theoretical tools around temporal relations that has already been applied to *multimodal systems* in [Nigay et al. 1993] [Jakkula et al. 2007] [Serrano et al. 2009] [Bellik 1995].

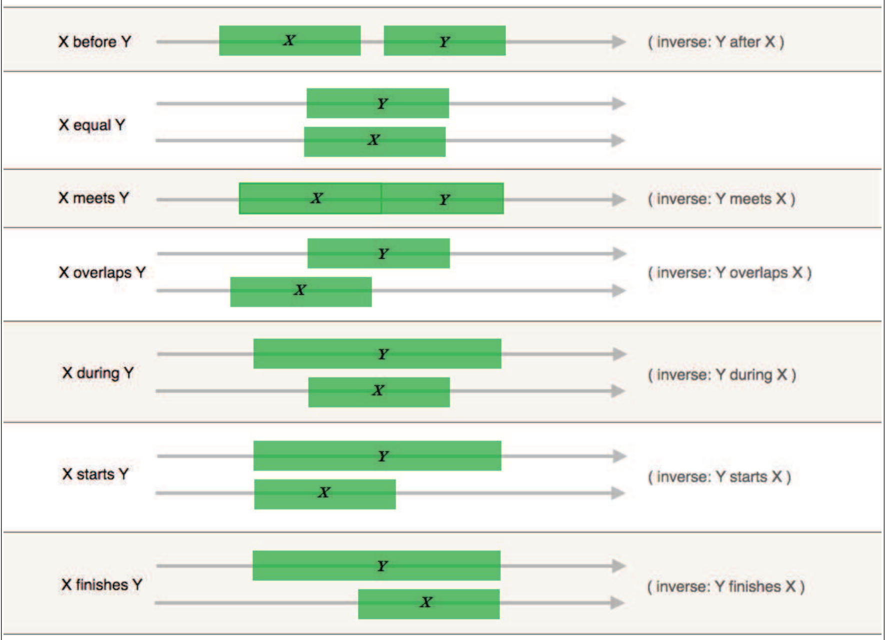


Figure 1.2.5: Allen's Temporal Interval Relations.

However, there are also other tools available to describe the temporal relationships between intervals that a current *multimodal system* could use to address temporal issues: the semi-interval relations of [Freksa 1992], the midpoint interval relations of [Roddick et al. 2005], the container relations of [Villafane 1999], the approximately equal relation of [Ultsch 1996] and the coincidence intersection of [Möerchen 2006a].

The semi-interval relations of [Freksa 1992] focuses on some current cases of incomplete or coarse temporal knowledge, e.g. when one interval boundary is unknown or when two relations between start or endpoints suffice to uniquely identify the relation. These are useful operators to express *symbolic time* relations [**Figure 1.2.6**] like older/younger, head to head, survives/survived by, tail to tail, precedes/succeeds or born before death/died after birth, by using incomplete data.

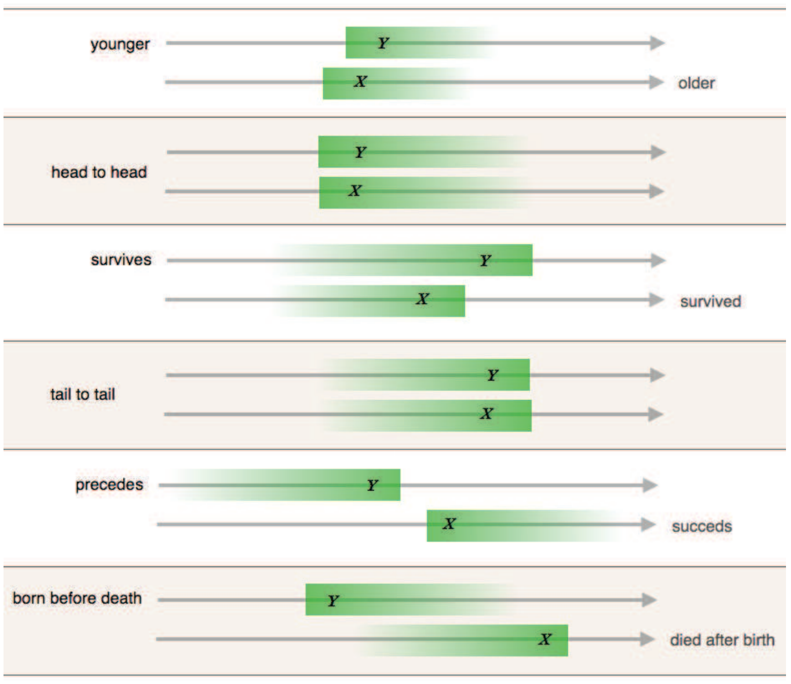


Figure 1.2.6: Semi-Interval Relations of Freksa.

In the midpoint interval relations of [Roddick et al. 2005] the Allen's relations are extended by taking into account the relation of each interval midpoint to the midpoint of the other interval. With this approach it is possible to focus on overlaps between intervals that can be largely overlapped or only overlapped to some extend [**Figure 1.2.7**]. As a result, 49 relations can be defined from Allen's relations, to handle coarse data with an arbitrary local order.

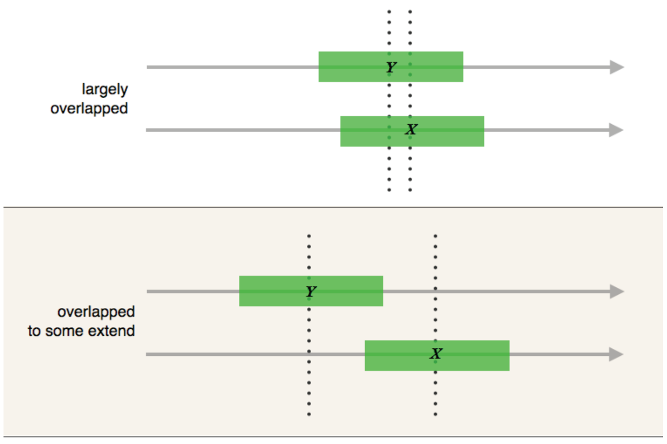


Figure 1.2.7: Midpoint Interval Relations of Roddick

[Villafane 1999] defines a generic operator to describe containment [**Figure 1.2.8**]. It uses some of the Allen's relations : equals, starts, during and ends. [Ultsch 1996] proposes a version of the equals operator of Allen that support approximation (aproximately equal): he calls it the «more or less simultaneous» operator [**Figure 1.2.8**]. In this operator the start and end point of the interval are not required to be exactly equal, just slightly different by a little interval of time. [Möerchen 2006a] proposes a coincides operator where he drops the constraints on boundary points requiring only some overlap between the intervals [**Figure 1.2.8**].

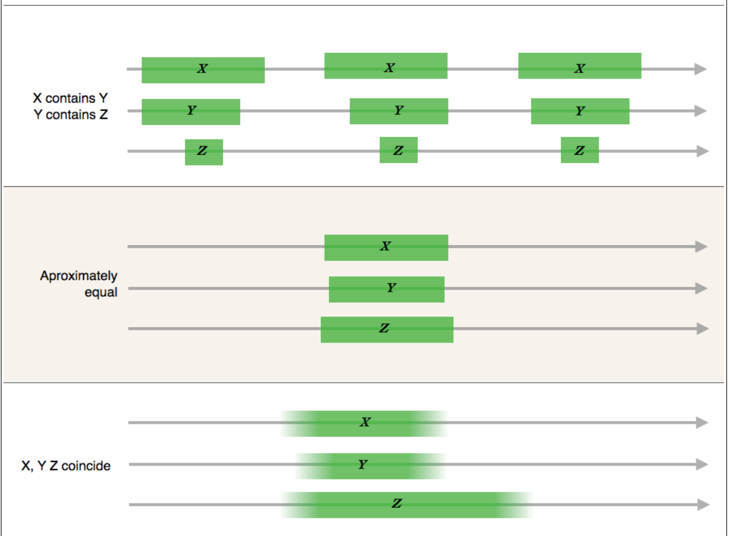


Figure 1.2.8: Villafane, Ultsch and Möerchen operators

With all these operators, the synchronization among modalities and their usage can be modeled following the semantic information about the specific feature to perform in accord with the application, the interaction and the temporal models.

In this way, if a current *multimodal system* wants to support *temporal* modeling, it could :

- Handle the **Time** differentiating the *time point*, the *time interval* or the *time semi-interval* granularity.
- Model the data distribution («sensed» / «returned») from an homogenous perspective and/or from an heterogeneous perspective.
- Model and differentiate the *time series*, the *time sequences*, the *interval series* or the *interval sequences* features.
- Manage the *valid time,* the *transaction time* or the *symbolic time.*
- Support the temporal relation aspects like *duration*, *periodicity*, *order*, *synchronicity*, *coincidence* and *concurrency.*
- Use temporal operators to describe and handle *time points*, *time interval* and *time semi-interval* relations.

1.2.3.3 Space-Time Situation

Most spatial models organize their information in order to describe the physical locations, the spatial relations between entities and the change of position of these entities on the Euclidian space.

They are fact-based models, describing a location with **geometric coordinates** that represent three physical informations -latitude, longitude and elevation- to locate a point in the space [Descartes 1637], or only two physical informations -latitude and longitude- to locate a point in a surface [Gauss 1827].

Nevertheless, in more relative contexts -as human action is-, **Time** cannot be separated from the three dimensions of space, because the observed rate at which **Time** passes for an object depends on the object's perception relative to the observer -in our case a *multimodal system* or a user- and also on the strength of proxemics[36], which can «slow» the «sensed» passage of **Time** and even intervene in the perception of the space-time itself.

Location in space is biased

In other words, the management of **Time** in a multimodal interaction can be affected by an «attentional bias» related to the location and the relationships between the participants. For this reason, a *multimodal system* can be forced to handle not only the temporal situation, but also the *space-time situation*.

With this goal, some space-time models can be symbolic. The **symbolic coordinates** use semantic identifiers that are socially defined, e.g. room numbers. They are appropriate as a topological model of relations between things or phenomena without explicit information about the physical space itself [Randell et al. 1989]. This representation capture the information in a way reflecting the **«semantic»** level. Simply put, the focus of the symbolic coordinates is the perceptual common sense.

Tools to model space-time situation

Nowadays, there are tools to model the *space-time situation* for spatio-temporal databases or spatio-temporal planning. One of them is the Ontology for Geographical Information System [Frank 2003] that is constructed from 5 tiers in an unified system.

The division of the ontology into tiers defines different levels of agreement between multiple data collections and in result, it integrates vector, raster and social data.

The tier 0 defines the «real» physical world, a four-dimensional continuous field of attributes values. The tier 1 handles the point of view of an observer (human or agent), the type of his observations -the raw data- and its measurement units, e.g. a coordinates system.

The tier 2 covers the abstraction of physical objects, defined by a perception based on affordances, uniform properties, topological relations, an identity that remains for the observer over some period of time and operators describing the changes in the object's identity (called its «lifestyle»).

The tier 3 includes the social reality in the sense of John Searle [Searle 1995]. For him, a football score or a room number are institutional facts that arise out of collective intentionality by convention but that are grounded in a physical reality of brute facts. For example, the driver's license number is the brute fact that indicates the institutional fact of having the social right of driving. For this tier of the ontology the most important premise is that «*social institutions are stable, evolve slowly and are not strongly observer dependent*». [Frank 2001]

These four first tiers are also represented in other spatial models like the Context Modeling Language [Henricksen et al. 2006] or the Nexus Project [Nicklas et al. 2001] as pointed by [Bettini et al. 2009].

Lastly, the tier 4 incorporates cognitive agents and their knowledge. It handles the individual or collective meta-data about the physical world, about its observations and about the social reality. This is the knowledge around our knowledge: precisely what the ontologies are. In this tier the Equator project [Millard et al. 2004] of interconnected symbolic spaces or the Situation and Event Model researches[37] can be considered as examples.

36 Proxemics theory argues that human perceptions of space, although derived from sensory apparatus are molded and patterned by culture. It is defined as «the study of man's transactions as he perceives and uses intimate, personal, social and public space in various settings» [Hall 1966] It is a kind of geographical spatial process which impacts the distribution of things in the space as «the tendencies for objects to come together in space (agglomeration) and to spread in space (diffusion)» [Getis et al. 1978] Proxemics studies also how man unconsciously structures micro space -the distance between men in the conduct of daily transactions, the organization of space in his houses and buildings, and ultimately the layout of his towns. It is a research based on the concept of territoriality, searching patterned distinctions while studying individual differences.
37 See supra part 1.2.3.1 Usage Situation

In addition to model the *space-time situation*, a *multimodal system* can need operators that describe the spatio-temporal relation between entities [Hallot 2006]. There are three types of spatial relations [Egenhofer 1989]: metric, ordinal and topological. 38

Ordinal and topological operators, have its emphasis on qualitative abstractions of spatio-temporal aspects of the perception, based on common-sense. Methodologically, these calculi restrict rich mathematical spatio-temporal theories to some specific aspects of these theories that can be treated with simple qualitative and non-metric languages. As we saw with temporal interval relations, these are abstract algebras of relations, for which the reasoning can be carried out at a symbolic level.

Qualitative operators to model space-time

While Allen's Interval Calculus [Allen et al. 1985] is the most well-known qualitative temporal calculus; on the spatial side, there is:

- the generalization of Allen's work to two dimensions [Balbiani et al. 1998],
- the topological Region Connection Calculus RCC-8 [Randell et al. 1992]

Combining the two kinds of representation and reasoning, there is:

- the Spatio-Temporal Constraints Calculus STCC [Gerevini et al. 2002],
- the Topology of space-time [Muller 2002]
- the Qualitative Trajectory Calculus [Van de Weghe et al. 2006].

These are spatial patterns that humans mentally manipulate over a time-ordered sequence of spatial transformations.

For example, the Region Connection Calculus describes regions in a geometric coordinates space or in a symbolic space by their possible relations to each other. RCC-8 consists of eight basic relations that are possible between two regions [**Figure 1.2.9**].

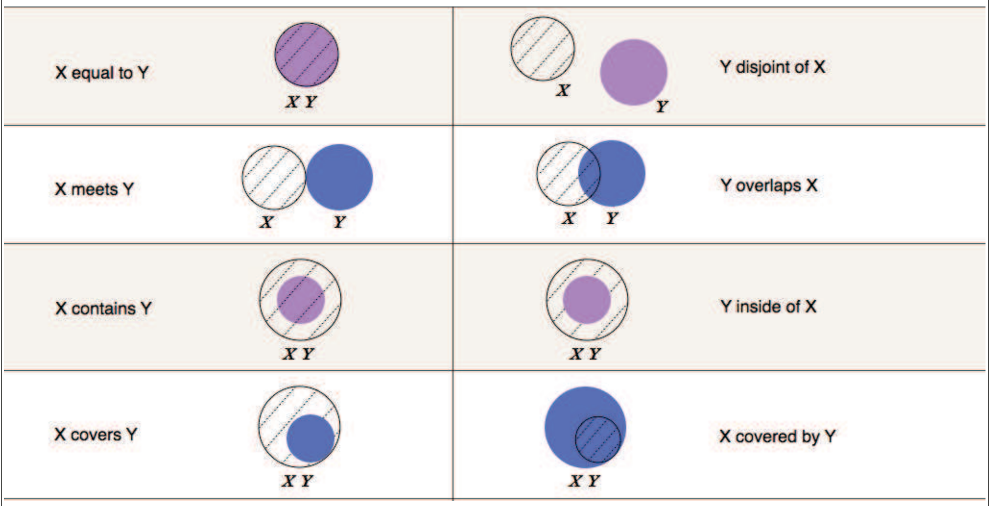


Figure 1.2.9: RCC-8 Relations.

The major advantage of qualitative relations to model the space in a *multimodal system* is that they are independent of specific values and granularities of representations. Depending on the situation and the granularity of the available knowledge, they can correspond to entities more specific or more generalized [Freksa 1992] as it is the case in the human space knowledge, which is mostly qualitative [Vieu 1997].

For example, we learn that Colombia is bigger than France. Our *primitive reference* is «greater than». Our first approach in learning is not the quantitative *positive reference* «the size of Colombia is 1,138,903 km2 and the size of France is 551,695 km2» but rather the *primitive reference* «Colombia is twice as larger than France» and this, depending on the context of the ongoing research or the learning process.

38 According to the New Oxford American dictionary, Topology is a major area of mathematics concerned with properties and spatial relations that are preserved under continuous changes in objects, for example in the neighborhood relation.

For this reason, qualitative reasoning works only with the essential information for the knowledge context represented by a small number of symbols like the «quantity spaces». [Kontchakov et al. 2007] proposes three values that represent a classification with reference to the value 0 and the two neighboring unbounded sets + and - to describe, for example, the relative location between two moving objects. Other representations in qualitative reasoning [**Figure 1.2.10**] are the cardinal relations -egocentric view- [Frank 1992] or the position of an entity described by several divisions of the space to build relationships with another entity in a grid. -allocentric view- [Ligozat 2006]

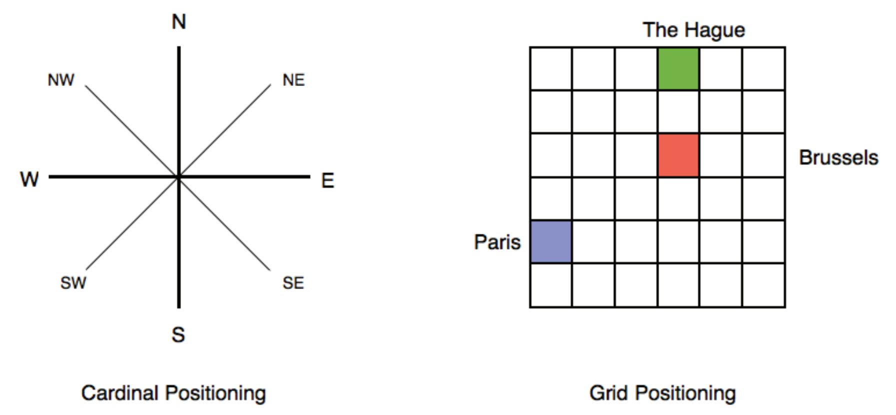


Figure 1.2.10: Spatial representations in qualitative reasoning.

Based on the grid, Ligozat asserts that «Paris is in the southwest of Brussels» and «Brussels is south of The Hague» which is a qualitative knowledge of spatial location. With this spatial knowledge, we can achieve inferences, e.g. «The Hague is located in the northeast of Paris». His example deals with directional knowledge, but it may also be topological, describe a form or explain a qualitative distance by defining the values that can make relationships, a set of operations applicable to these relations, and finally a set of axioms defining the results of these operations.

By their «naïve»[39] nature, this representations generally reflect natural language [Haspelmath 2006]. According to Frank, the latter uses a location description method with respect to spatial frames of reference present in each language. [Frank 1998] He proposes definitions for these methods on English and a method to extend the analysis to other languages. They are:

Spatial frames of reference

- For Absolute Reference Systems : In this systems the orientation is given from the outside, a cardinal direction, inland/seaward, up-down a landmark direction. They possesses two frames of reference: egocentric «the ball is to the west» (centered in the speaker with cardinal orientation) and allocentric «the ball is to the west of the field» (centered in an entity with cardinal orientation).

- For Relative Reference Systems : In this systems the orientation is body-centered, as in front/back, left/right cases. They possesses two frames of reference: egocentric «the chair is before me» (centered in the speaker with egocentric orientation), intrinsic «the chair is in front of the mirror» (centered in an oriented entity with the ground entity orientation), and retinal or deictic «the coin is left of the ball» (centered in an entity with speaker-related orientation).

39 Naïve physics is a large-scale formalism of commonsense knowledge about the world: «I propose the construction of a formalization of a sizable portion of common-sense knowledge about the everyday physical world: about objects, shape, space, movement, substances (solids and liquids), time, etc.[...] Consider the following collection of words: inside, outside, door, portal, window, gate, way in, way out, wall, boundary, container, obstacle, barrier, way past, way through, at, in. I think these words hint at a cluster of related concepts which are of fundamental importance to naïve physics. This cluster concerns the dividing up of three-dimensional space in pieces which have physical boundaries, and the ways in which these pieces of space can be connected to each other, and how objects, people, events and liquids can get from one such place to another.» [Hayes 1978]

The qualitative representation in an egocentric frame of reference [Frank 1998] as illustrated in [**Figure 1.2.11**] differentiates not only the cardinal directions but also 4 distance relations -named zones-, where each successive range reaches twice as far as the previous one. The zone 1 unit is ***here***, the zone 2 unit is ***near***, the zone unit 3 is ***far*** and the latest not limited zone is ***very far***.

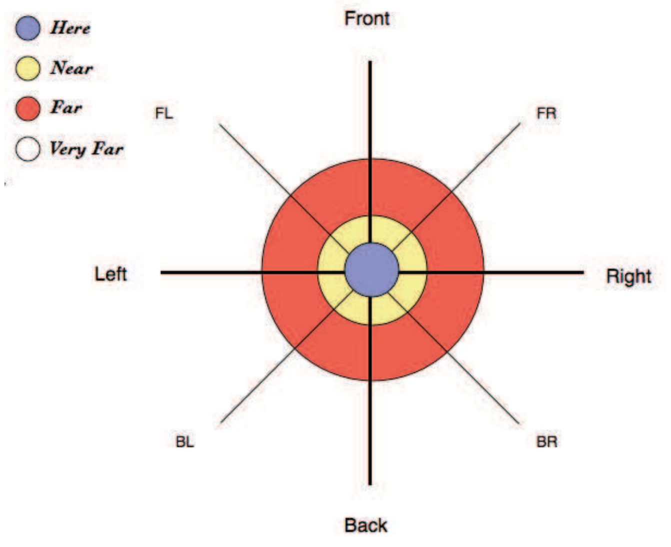


Figure 1.2.11: The egocentric qualitative distances

These zones correspond to the interpersonal distances proposed by Hall in the proxemics framework [Hall 1966]: the intimate, personal, social and public spaces. Following this framework, in a *multimodal system*, increasing distances will result in degraded thermal, olfactory, visual and aural communication and perception between interactors [Walters et al. 2005a]. For this reason the zone ranges given by proxemics research, can be taken into account. [Walters et al. 2005b] [Pacchierotti et al. 2006] They are summarized in [**Table 1.2.3**] from [Lambert 2004] :

Personal Spatial Zone	Range	Situation
Close Intimate	0 to 15 cm	Touch, Olfaction (lover or close friend)
Intimate Zone	15 cm to 45 cm	Touch (lover or close friend only)
Personal Zone	45 cm to 1,2 m	Speak (friends)
Social Zone	1,2 m to 3,6 m	Speak (non-friends)
Public Zone	3,6 m and more	Speak (public)

Table 1.2.3: Human-human Personal Zones

Nevertheless, the proxemics behavior is affected by a large range of social and cultural factors which makes its conceptual use very dependent on non formal researches and reliant on ad-hoc implementation. Probably for this reason, only a few recent works using social signal processing have tried to apply proxemics in multimodal systems. According to [Cristani et al. 2011] it is present mainly in smart meeting rooms [Gatica-Perez 2009] [Vinciarelli et al. 2009] and robotics [Michalowski et al. 2006] [Pachierotti et al. 2005] [Nakauchi et al. 2000].

Another field using proxemics is the analysis of spatial process[40] [Hofer et al. 2009] in studies about the dynamics of people moving through public spaces with real-time crowds models [Thalmann et al. 1999] [Musse et al. 1997] inspired on fluid dynamics and particle physics models.

40 «...spatial process are process taking place in space and may depend on location in space» [Hofer et al. 2009]

For example, by using formal abstractions as the Social Force Model [Helbing et al. 1998], researchers as [Pellegrini et al. 2009] [Scovanner et al. 2009] are modeling repulsive and attractive phenomena for multi-human pedestrian tracking.

With tools like the Ontology for Geographical Information System or the Spatial Logics [Varzi 2007] and the arsenal of operators provided by the qualitative reasoning and proxemics researches, the instantiation of modalities and their synchronization based on spatio-temporal semantics, can be modeled today in a *multimodal system* in order to handle the interaction and usage situations.

Thus, if a *multimodal system* need to support *spatio-temporal* modeling, it could :

- Interpret the spacial information with geometric or symbolic coordinates.
- Handle egocentric or allocentric representations of the space.
- Take in to account the emotional and social «attentional bias» that affects the perception of the space and the multimodal interactions on the space in a **semantic** level.
- Take in to account the metric, ordinal or the topological spatial relations.
- Model the *space-time situation* with ontologies or context description languages.
- Use spatio-temporal operators describing phenomena based on common-sense.
- Manage qualitative spatial relations like the «quantity spaces», the cardinal relations or the grid-based relations.
- Take in to account the proxemics distance zones and spatial process dynamics.

In this section we studied some attributes related to the delivery context of a *multimodal system*, and in particular, the use of some modeling tools to handle ***for what?***, ***when?*** and ***where?*** a *multimodal system* can operate.[41]

An example of the link between these notions and their use on multimodal systems can be the illustration of the use-case presented in Appendix 6. As we already presented, a multimodal context can be described as the relationship between devices, users and a given domain (applications, modalities, interaction and use) with a precise intention in a given space-time.

Then, in our use case example, the intention is to see the news in a given space-time: a cooking situation. One device and four servers (distant devices) are related with the final user through a given application, composed by a dynamic number of modalities depending on the interaction and the type of use.

The application is an HTML5 web page displaying a video player. The modalities are a speech recognizer, an image recognizer, a Text to Speech synthesizer and a HTML GUI. The interaction is each step composed of the user and system actions. The type of use is the way the user interacts, in our case, a distracted way, given its parallel activity of cooking. The whole context is in this way described by a combination of the characteristics described in this section, and their dynamic relationships.

Depending on the nature and complexity of the system implementation, it can posses one, many or even none of these attributes because, for example, working with a robot will require a different kind of usage situation model and spatio-temporal model than the one needed for a multimodal mobile application.

Anyhow, based on these situation modeling possibilities, we can classify *multimodal systems* according to the presence of some of these attributes **[Table 1.2.4]** :

41 This is the third part (*Which*) of a description of a distributed multimodal system characteristics according to the knowledge model proposed in this document on section 2.2.3.2 A Knowledge Model for Multimodal Discovery.

Usage Situation	**At Model Level**		**Relationships**	
	At Profile Level		**Mental Models**	
	Factual Description		**Social Qualities**	
	Interpretation Description			

Temporal Situation	**As Points**		
	As Intervals		
	As semi-Intervals		
	Sensed Data Distribution	Homogeneous	
		Heterogenous	
	Returned Data Distribution	Homogeneous	
		Heterogenous	
	Relational Aspects		

Space-time Situation	**Ontologies**			**Emotional Bias**		
	Other Context Descriptions			**Social Bias**		
	Coordinates	Geometric		**Proxemic**	Distance	
		Symbolic			Dynamics	
	Relations	Metrical		**Qualitative Relations**	Quantity Spaces	
		Ordinal			Cardinal	
		Topological			Grid-Based	
	Views	Allocentric		**Naïve Operators**		
		Egocentric				

Table 1.2.4: Delivery Context and Situation factors for the analysis of Multimodal Systems.

To complete our exploration about what a *multimodal system* is today, we have presented in this section a standpoint for the classification of multimodal systems. This allowed us to extend the initial definition given in sections 1.1 and 1.2 with some attributes -about behavior, participants and situation of delivery- which can be reflected in the final architecture of a *multimodal system* but also that can be studied as possible architectural building blocks for a generic Architecture of Reference.

With this goal, it is important to check in real multimodal implementations and architectures how many of these building blocks are present and how they are designed. This will be the aim of the next section.

1.3 Architectures in Recent Multimodal Systems.

In this section we will use the classification proposed[42] as a frame of reference to pass across real implementations of multimodal architectures.

According to [Lalane et al. 2009] the multimodal researches around fusion engines would reached the maturity level of the BRETAM model [Gaines 1990].

In this linear model, a technological product progresses through six phases. First comes a **B**reakthrough , involving creative ideas produced in a trial and error environment, then the ideas are **R**eplicated and validated by other researchers.

Then follows a period of **E**mpiricism in which the experience gained with the ideas results in some design rules. This leads to the development of underlying **T**heories, which eventually are applied in a more **A**utomatic way. Finally, the technology reach its **M**aturity adopted in mass production.

Position of the Multimodal Architecture research in the BRETAM model

Though, the research around the standards for Multimodal Architectures doesn't seem to follow the same rhythm of evolution. Since the year 1970 until approximately 2003, ideas were produced from trial and error, and these ideas were adopted by multiple researchers. Yet, until the multisided initiative at the W3C (international industrials and researchers), the various experiences of implementation and theoretical proposals of multimodal architectures lacked of commonly accepted and standardized design rules that could lead to a shared Theory and a more Automatic application of principles of architectural design for multimodal systems.

For this reason, this section presents the state of the art of the most relevant multimodal architectures proposed by the research community with the idea of an empirical analysis of the similarities between the implementations of multimodal systems in order to confront them with the W3C's proposal and eventually, enrich the latter.

Thus, this enumeration will be made under the light of the the Multimodal Runtime Framework proposed by the W3C. To the best of our knowledge, this Framework is the only theoretical and generic model of a Multimodal Components Architecture intended to be applied in a standardized way[43] to reach the maturity needed for the mass production and interoperability of multimodal applications in large-scale multi-domain networks.

Outline

With this goal, first, we will present the Multimodal Runtime Framework & the Multimodal Architecture and Interfaces recommendations; secondly we will discuss the most relevant multimodal architectures according to the requirements given in section 1.2. Finally, we will evaluate the Multimodal Runtime Framework model in opposition to the concrete implementations studied.

1.3.1 Architectures from Open Standards

The user interaction with applications on mobile phones, personal computers, tablets or other electronic devices is moving towards a multi-mode environment in which important parts of the interaction are supported in multiple ways. This heterogeneity is driven by applications that compete to enrich the user experience in access to all kinds of services.

More and more, applications need interaction variety, which has been proven to provide numerous concurrent advantages (e.g the user rich experience as part of the iPhone breakthrough innovation in the mobile market). At the same time, it brings new challenges in multimodal integration, which is often quite difficult to handle in a context with multiple networks and input/output resources.

Today, users, vendors, operators and broadcasters can produce and use all kinds of different media and devices that are capable to support multiple modes of input or output. In this context, tools for authoring, edition or distribution of media are very mature as proprietary / open source tools and services that handle, capture, present, play or recognize media in multiple modes.

42 See supra part 1.2. Definition of a Multimodal System

43 Even if [Dumas et al. 2008] presents a canonical architecture based on 4 components their intent was not to propose an architecture of reference or to collaborate in the definition of a standard, but to analyze current usages in design of multimodal architectures.

Nevertheless, there is a lack of powerful practices to structure in an architecture that can enhance the integration and synchronization of all these media in a interoperable and standardized way. In other words, distributed multimodal architectures are in the phase of the BRETAM model in which design rules coming from the Empiric experience are proposed.

Today, there is no interoperable way to build an application that uses internet technologies for dynamically combine and control discovered modalities. The Multimodal Architecture and Interfaces recommendation address this problem.

The «Multimodal Architecture and Interfaces» [W3C-MMIA 2012] is an open standard developed by the World Wide Consortium since 2005. Currently (2012) it is a working draft of the W3C's Multimodal Interaction Working Group. The document is a technical report specifying a multimodal system architecture and its generic interfaces to facilitate the integration and the management of the multimodal interaction in a distributed system.

The Multimodal Architecture and Interfaces is the specified description of a larger infrastructure called «The Multimodal Runtime Framework» [W3C-MMIF 2003] which recommends guidelines about the main functions that a multimodal system can need [**Figure 1.3.1**]. This framework is at a higher level of abstraction than the MMI Architecture and Interfaces recommendation. The MMI Runtime Framework is the runtime support and communication modules of the multimodal system while MMI Architecture is the description and the specification of its main modules, its interfaces and its communication modes.

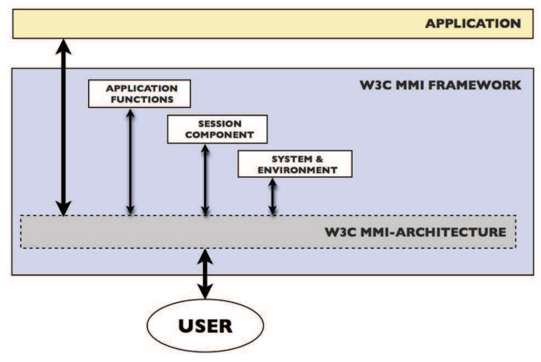


Figure 1.3.1: The place of the MMI Architecture in the MMI Framework

Purpose of the MMI Framework

The purpose of the MMI framework is to identify the major components that every multimodal system could need as a set of related functions [**Figure 1.3.2**] and the markup languages needed to describe the information required by components and for data flowing among components:

- handlers for input/output participants to the interaction represented in this research with the color code **green** and named Modality Components in the W3C's recommendation,

- a coordination (orchestration) component for the interaction modalities, represented with the color code **orange** and called Interaction Manager in the W3C's recommendation,

- a component responsible of data handling represented with the color code **violet** and called the Data Component.

- a component responsible of following the state of the components and of the handling of the multimodal session. This kind of component is represented with the color **turquoise**. In the MMI framework it is called the Session Component.

- a System and Environment Component to handle the context state also represented in this research with the color **turquoise** because for us, the context state is part of the multimodal session state.

- a component responsible to handle the application functions, represented in this research with the color code **brown**.

- and a transport layer for the interaction events, represented in this research with the color code **light blue**.

Relation between the MMI Runtime Framework and the MMI Architecture and Interfaces

By its purpose, the Multimodal Interaction Framework is not an architecture , since for the W3C Working Group an architecture indicates how components are allocated to hardware devices and the communication system enabling the hardware devices to communicate with each other.

For this reason, the «Multimodal Architecture and Interfaces» recommendation completes this framework by introducing a generic structure of architectural modules and a communication protocol. It is an event-driven architecture proposed as a general frame of reference for the exchange of control flow data in multimodal systems.

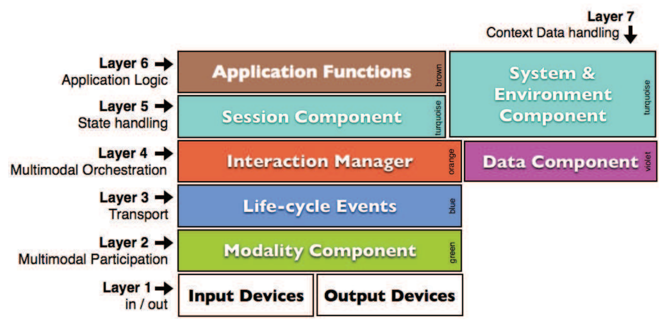


Figure 1.3.2: The MMI Framework & Architecture Layers.

It can be used to determine the basic infrastructures and layers needed to command the multimodal features provided by the user interface in applications. This architecture is also proposed to facilitate the task of implementing several types of multimodal service providers on multiple devices: mobile devices and cell phones, home appliances, Internet of Things objects, television and home networks, enterprise applications, web applications, smart cars or on medical devices and applications.

1.3.1.1 The Interaction Model in the MMI Architecture

The Multimodal Architecture and Interfaces specification is based on the MVC design pattern [**Figure** 1.3.3] developed for the Smalltalk platform in the late 1970s [Goldberg 1984] refined by some of the proposals of the PAC architecture model [Coutaz 1991]. The result organizes the user interface structure in three parts: the Model, the View and the Controller.

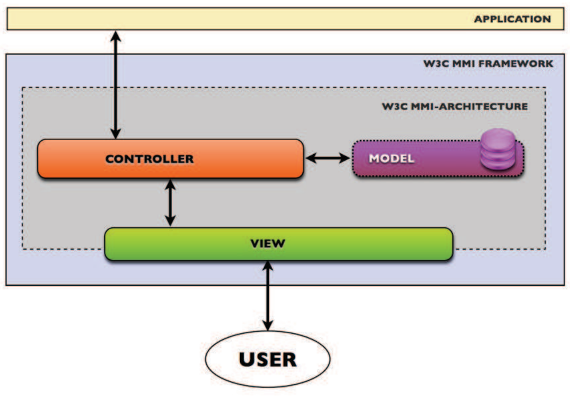


Figure 1.3.3: The MVC & PAC Models in the MMI Architecture

The MVC model in the MMI Architecture

A particularity of the MVC model applied in the MMI Architecture is that the presentation layer generalizes the View part to the broader context of the multimodal interaction, where the user can use a combination of visual, auditory, biometric and / or tactile modalities. The architecture separates the application logic of the user interface description by a mediator: the controller. It allows to aggregate modality dependent processing, facilitates the addition of new modalities and enables the timing-sensitive control of modalities.

The Multimodal Architecture and Interfaces specification is compliant with the MVC and the PAC design patterns for multimodal architectures, by keeping the control of the interaction in an unified way within the control layer.

A software architecture design pattern (or reference model) is a standard decomposition of known systems into functional components coupled in a well-defined way, providing a generic solution. The MVC and the PAC are two multi-agent approaches which explicitly deal with fine-grain modularity and parallelism.

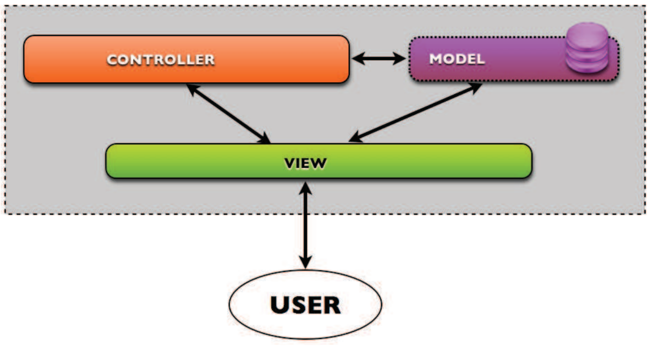


Figure 1.3.4: The MVC architecture design pattern

In the MVC model [**Figure 1.3.4**], the Controller translates the user's actions into method calls on the Model. The Model broadcasts a notification to the View and to the Controller to inform that its state has changed. The View queries the Model to determine the exact change. Upon reception of the response, the View updates the display according with the information received. Thus, in the MVC patter, the View is directly linked with its controller but it can also query and communicate with the Model. 44

In the MVC pattern [**Figure 1.3.5**], the Model offers a registration mechanism so that multiple Views and Controllers can express their interest in the Model through anonymous callbacks. This allows an easy implementation of multiple renderings of the same domain concepts either on the device or across multiple distributed devices. It also allows parent or related Views, Controllers or Models to communicate with each other.

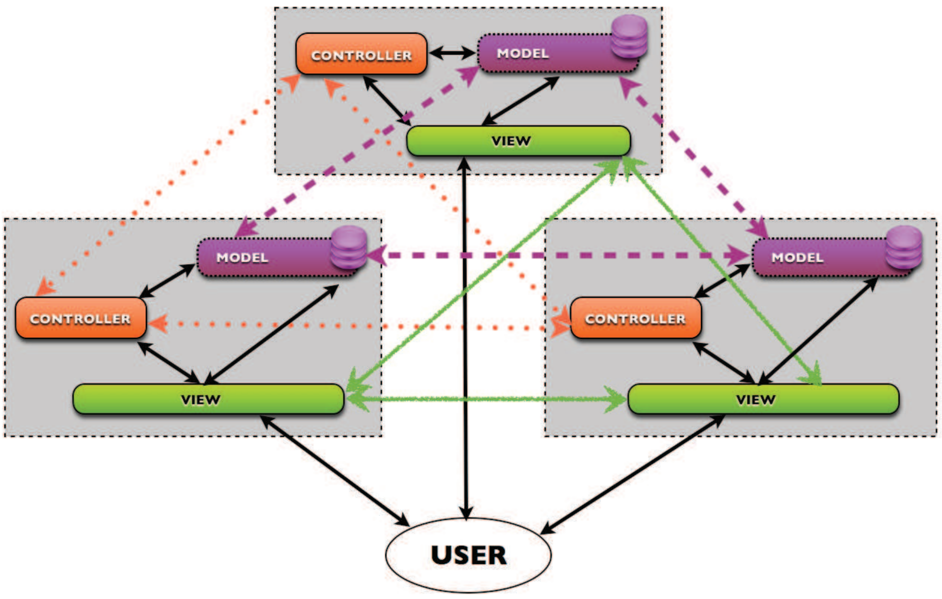


Figure 1.3.5: The MVC communication policy

44 For a detailed description of the MVC communication mechanisms, see: Applications Programming in Smalltalk-80(TM)] at http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

In the PAC pattern [**Figure 1.3.6**], an agent has a Presentation (i.e., its perceivable input and output behavior), an Abstraction (i.e., its functional core), and a Control facet (ie. its communicating hub and its mediator between presentation and abstraction facets).

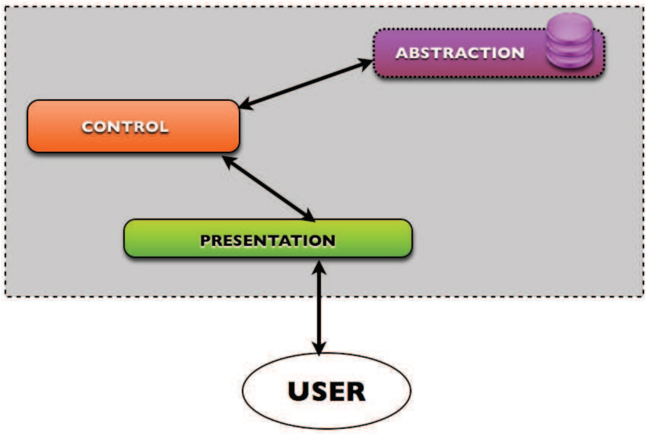


Figure 1.3.6: The PAC architecture design pattern

In the PAC pattern, no agent Abstraction is authorized to communicate directly with its corresponding Presentation and vice versa. Communications of any sort are conveyed via Controls that serve as a glue mechanism to express coordination and transformations between the abstract (Model) and the concrete (Presentation) perspectives. In this way, the flow of information transit through the Controls in a hierarchical way and in distributed environments, Controls communicate together to maintain the interaction coherence and dependencies [**Figure 1.3.7**].

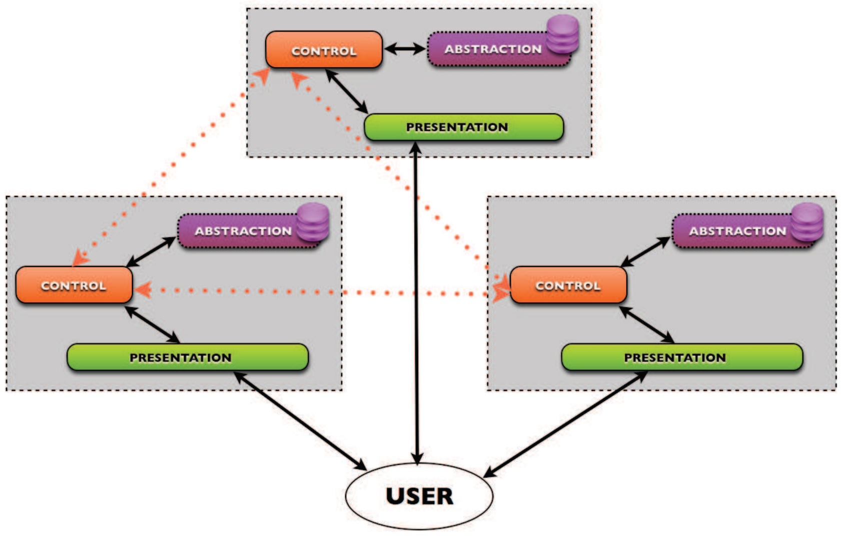


Figure 1.3.7: The PAC communication policy

In the MMI architecture, the changes in the Modality Component (which represents the MVC view or the Presentation facet of PAC agents) are commanded from the control layer. The Interaction Manager translates the user's actions into method calls on the Data Component, like the MVC pattern proposes. But also, the Modality Component's communication and request of information is restricted to exchanges with the Control layer as the PAC pattern defines. The Model broadcasts a notification to the Interaction Manager, who commands the Modality Component to change using one of the interaction life-cycle events. Upon reception of the event, the Modality Component updates the user interface according with the information received.

Another characteristic of the architecture is its recursion [**Figure 1.3.8**]. The modules are black boxes and it is possible to encapsulate several components in a more complex component, which communicates with an Interaction Manager at a higher level. In this way, the architecture follows the nested dolls principle.

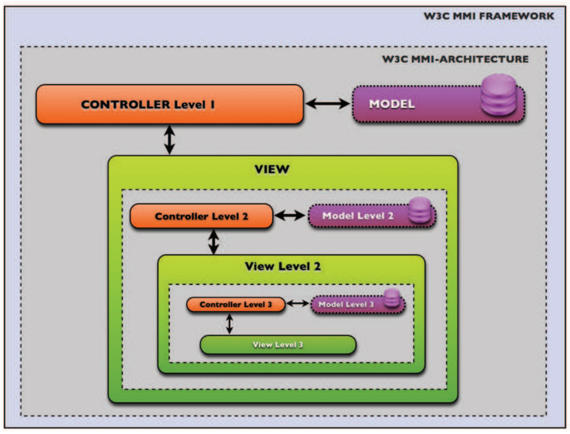


Figure 1.3.8: Recursion in the MMI Architecture Structure

One focus of the MMI Architecture is the distributed use cases

The specification address also the distribution issues: its goal is to facilitate the implementation on multiple material resources in a network or a centralized implementation with all the modules installed in a single material support. The sharing of information between modules is loose coupled. This promotes low dependence between modules, reducing the impact of changes in one module on other modules, and facilitating their reuse. In this way, the modules have little or no knowledge of the functioning of any other module, and the communication between them is done through the exchange of messages, following a precise communication protocol provided by the Architecture's API.

1.3.1.2 The MMI Architecture Modules

Basic components of the MMI Architecture and Interfaces

The MMI Architecture and Interfaces recommendation distinguishes three types of components [**Figure 1.3.9**]: the Interaction Manager, the Data Component and the Modality Components.

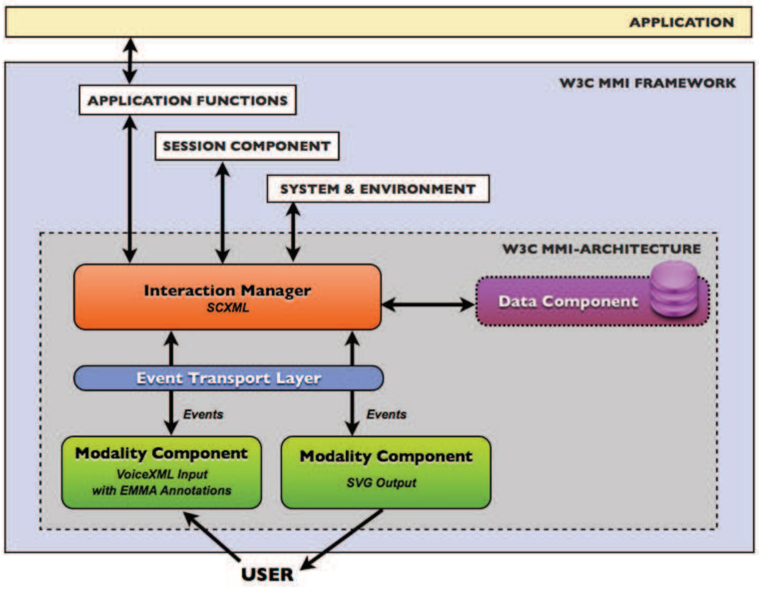


Figure 1.3.9: Modules of the MMI Architecture in the MMI Runtime Framework

- The Interaction Manager

The Interaction Manager's responsibilities

The Interaction Manager is a logical component, responsible for all the exchanges of messages between the components of the system and the multimodal Runtime Framework. It is a communication bus and also an event handler. Each application can configure at least one Interaction Manager to define the required interaction logic. This controller is the core of the multimodal interaction:

- It manages the specific behaviors triggered by the events exchanged between the various input and output components.
- It manages the communication between the modules and the client application.
- It ensures consistency between multiple inputs and outputs and provides a general perception of the application's current status.
- It is responsible for data synchronization.
- It is responsible for focus management.
- It manages communication with any other entity outside the system.

- The Modality Components

The Modality Component's responsibilities

The Modality Components are *Sensor Systems*[3] and *Effector Systems*[4] responsible for specific tasks, including handling inputs and outputs in various ways, such as speech, writing, video, etc..

According to the MMI Runtime Framework and the MMI Architecture and Interfaces specifications, the Modality Components :

- Can manage the input commands and the input recognition.
- Can manage the semantic interpretation of recognized inputs.
- Can manage the integration of inputs (*data fusion, feature fusion, semantic fusion* or *hybrid fusion*)[45]. **[Figure 1.3.10]**
- Can manage the generation of the output content (*semantic fission, modality fission* or *data fission*)[46].
- Can manage the styling and adaptation of content.
- Can manage the rendering of content.

These are logical entities that handle the input and output of different hardware devices (microphone, graphic tablet, keyboard) and software services (motion detection, biometric changes, 3D rendering, spatial sound reproduction) associated with the multimodal system.

Modality Components are participants on the multimodal coordination(orchestration) directed by the Interaction Manager.

A Modality Component can potentially wrap multiple features provided by multiple physical devices but also more than one Modality Component could be included in a single device. **[Figure 1.3.10]**

To this extent the Modality Component is an abstraction above the device and the device driver used to add multimodal handling and processing that can be implemented differently in each case .

45 See supra part 1.1.6. Fusion
46 See supra part 1.1.7. Fission

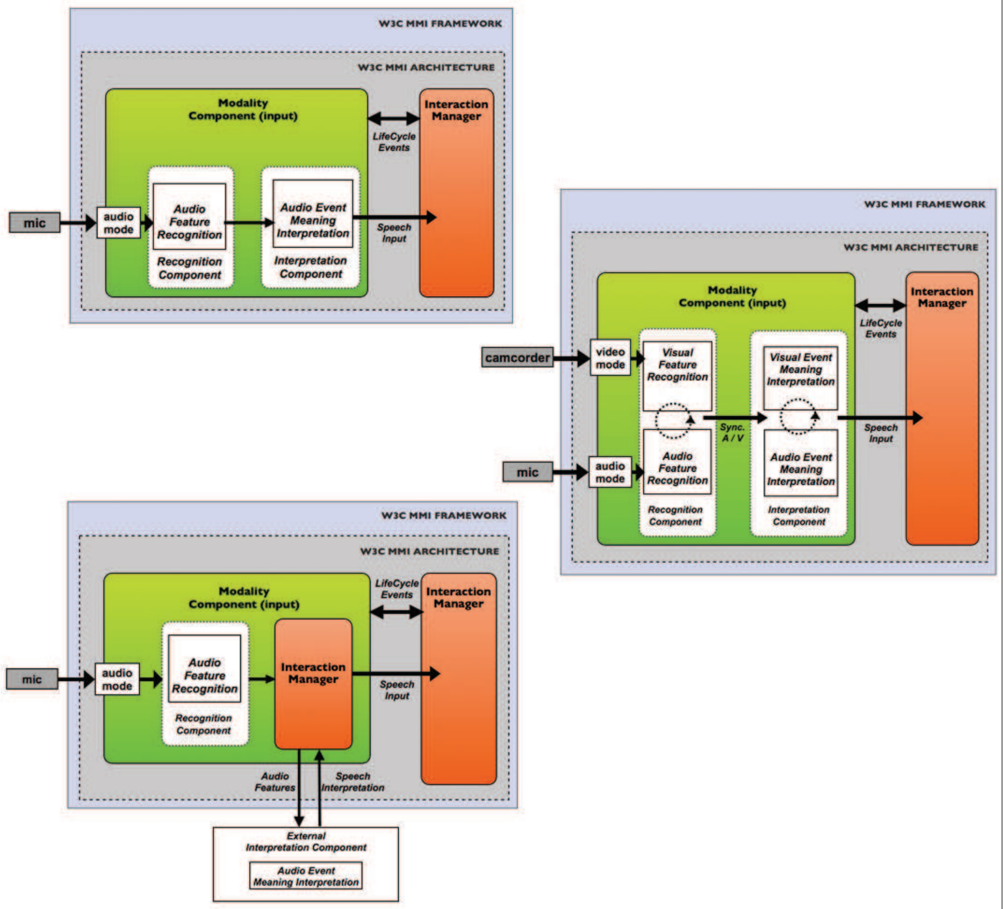

W3C MMI FRAMEWORK
W3C MMI ARCHITECTURE
Modality Component (input)
Interaction Manager
LifeCycle Events
mic
audio mode
Audio Feature Recognition
Recognition Component
Audio Event Meaning Interpretation
Interpretation Component
Speech Input
camcorder
video mode
Visual Feature Recognition
Visual Event Meaning Interpretation
Sync. A / V
Audio Event Meaning Interpretation
Audio Features
Speech Interpretation
External Interpretation Component
Audio Event Meaning Interpretation

Figure 1.3.10: MMI Modality Components as Sensor Systems for devices

For this reason, the W3C recommendation currently (2012) does not describe in detail the structure or implementation of the Modality Components. It focuses only on the need of a communication interface with the Interaction Manager and the need of an implementation that follows a specific communication protocol.

- The Data Component

The Data Component's responsibilities

The primary role of the Data Component's is to save the data of the application that may be required by one or several modality components or by other modules (e.g, the Framework's session module).

Only the Interaction Manager has direct access to the Data Component and it is the only component that can view and edit the data or communicate with external servers if necessary. As a result, the Modality Components must use the Interaction Manager as an intermediary to access the data of the multimodal application.

In addition, for the storage of private data, each Modality Component can implement its own Data Component [**Figure** 1.3.11]. This nested Data Component keeps the data that the Modality Component may require, for example, in speech recognition tasks. In this case, the Modality Component must be implemented in a nested way according to the nested Dolls Principle proposed by the MMI Architecture and Interfaces Recommendation [**Figure** 1.3.8]. As a result, the specification proposes two categories of Modality Components : a) the simple Modality Component without a nested Interaction Manager or Data Component and b) the complex Modality Component, composed of a nested Interaction Manager and eventually, a nested Data Component.

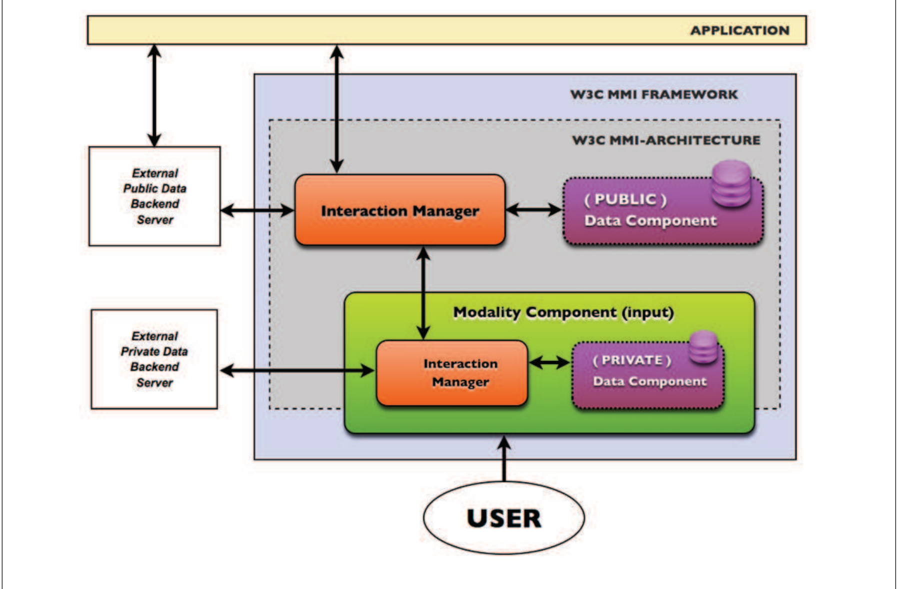

APPLICATION
W3C MMI FRAMEWORK
W3C MMI-ARCHITECTURE
External Public Data Backend Server
Interaction Manager
(PUBLIC) Data Component
Modality Component (input)
External Private Data Backend Server
Interaction Manager
(PRIVATE) Data Component
USER

Figure 1.3.11: Distribution of Data Components

1.3.1.3 The MMI Communication Protocol

In the MMI architecture, the communication protocol is asynchronous, bi-directional and based on the exchange of event notifications that are raised by the system following a user action or some internal activity.

This protocol defines the mode of exchange and how to establish and end the communication between modules with the Life-Cycle Events. In [**Figure 1.3.12**] this events are represented with a color code that corresponds to the type of component responsible of triggering the event, for example, only the Interaction Manager can trigger the start event or the cancel event.

The role of the Life-Cycle events.

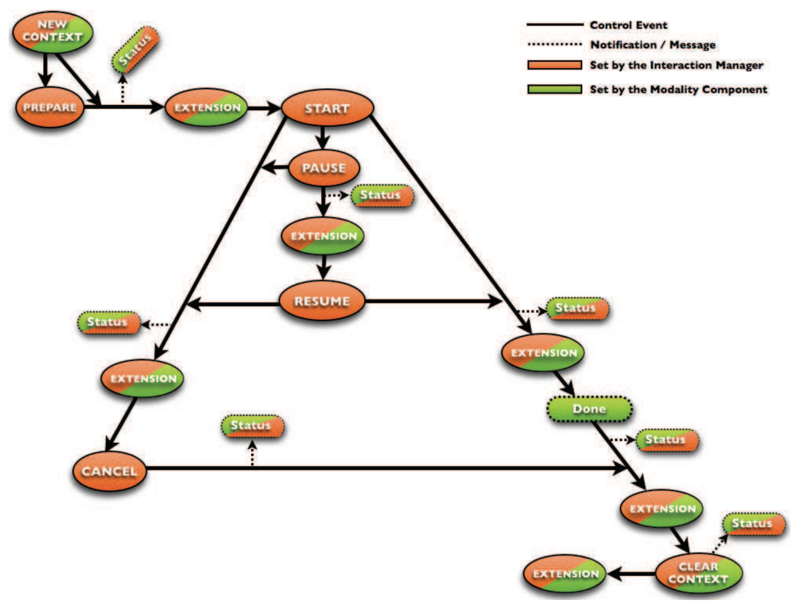


Figure 1.3.12: Multimodal Interaction Life-Cycle Events

These Life-Cycle Events are six standard events of control that are proposed to command devices and material services (such as a video player, a recognizer or a sound reproduction device) and two notifications proposed to monitoring the current status of the Component. The six standard Life-Cycle Events are specified as pairs of Request > Response exchanges:

Interaction Life-cycle events : Six pairs Request>Response Events

- NewContext (NewContextRequest / NewContextResponse)

Indicates the creation of a cycle of interaction (context) between zero, one or more users with one or multiple modality components.

Definition of Context in the MMI Architecture

The context is the longest period of interaction in which the modules must keep the information available. The context is associated with the semantics of the user's and system's activity during the interaction cycle. This allows the implementation to decide whether to keeping the information still makes sense for the execution of the current activity. This definition of context refers is different to our definition of interaction context, given in section 1.2.3 The Interaction Context of a Multimodal System that refers mostly to the non-technical environment surrounding the user interaction.

Usually the context is created from user input. The event is normally sent by one or more modality components to the interaction manager that must answer the query. For example, a modality component responsible for managing the inputs associated with the finger gesture (touchmove) in a web page viewed on a touch screen. At the beginning of the physical interaction, the modality component will send the query:

```
<mmi:mmi xmlns:mmi="http://www.w3.org/2008/04/mmi-arch" version="1.0">
<mmi:newContextRequest requestID="myReq1" source="myPointerMC.php" target="myIM.php" data="myMCStatus.xml" />
</mmi:mmi>
```

At this request the Interaction Manager will respond:

```
<mmi:mmi xmlns:mmi="http://www.w3.org/2008/04/mmi-arch" version="1.0">
<mmi:newContextResponse  requestID="myReq1"  source="myIM.php"  target="myPointerMC.php"  context="myContextID1"
status="success" />
</mmi:mmi>
```

With this exchange, the interaction cycle is officially declared and started for the system and the state of the components involved normally must change. Thus, the semantics of this event is the initialization of a new context of communication between participants.

- ClearContext (ClearContextRequest / ClearContextResponse)

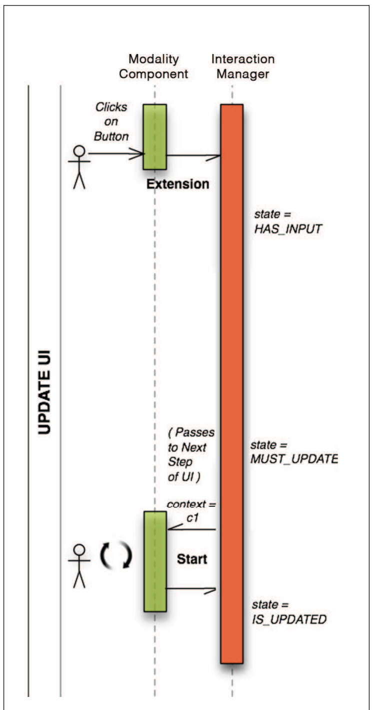


Figure 1.3.13: the Start Event

Sent by the Interaction Manager, the event marks the end of the cycle of interaction (context) and request the freed of the resources that were allocated to the current interaction context.

- Prepare (PrepareRequest / PrepareResponse)

Sent by the Interaction Manager, this event control signals to the modality component that it must be prepared to start its task and that it can load data that will be required to perform its work.

If there are multiple documents or data to be loaded during the preparation phase, the Interaction Manager could trigger the PrepareRequest event several times without starting the task after each request. Nevertheless, each request call must be answered.

- Start (StartRequest / StartResponse)

Sent by the Interaction Manager this control event signals to the Modality Component that it may start its task [**Figure 1.3.9**]. If during the execution of its work the Modality Component receives a new StartRequest event it may either start the new task, or report the failure.

- Cancel (CancelRequest /CancelResponse)

Sent by the Interaction Manager, this control event signals to the Modality Component that it should stop its current task.

- Pause (PauseRequest / PauseResponse)

Sent by the Interaction Manager, this control event signals to the Modality Component that it must pause its current task.

- Resume (ResumeRequest / ResumeResponse)

Sent by the Interaction Manager, this control event signals to the Modality Component that the task previously paused should resume.

- Status (StatusRequest / StatusResponse)

It is sent either by the Interaction Manager or by the Modality Components. This event indicates whether the cycle of interaction is still valid, that is, if the context is "alive".

The specification also recommends two types of notifications, which one, the Extension Notification, can contain control or command data to handle devices or material services. For this reason this notification is regarded like an exception: this is a standard control event that is not described as a pair Request > Response. These two notifications are:

Interaction Life-cycle events : Two Notifications

- Extension (ExtensionNotification)

This event is used to communicate application-specific control data or any other needed data. It can be generated either by the Interaction Manager or by a Modality Component [**Figure 1.3.9**]. It ensures the extensibility of the architecture by giving a generic API dedicated to the application-specific needs. For example, a Modality Component that handles a DVD player may indicate to the system an interaction with the DVD's main menu.

- Done (DoneNotification)

This notification is sent by the Modality Component to the Interaction Manager when it has finished its task.

For example, if a photo album application starts a procedure to recognize people in a photo album, it can demand to an image recognizer modality component available in the system a recognition task, and then continues with the user interaction, for example, showing random images or an audio waiting message. When a recognizer Modality Component has finished the image recognition task (finding a face), it will send the doneNotification, with the result to the interaction manager, who finally can render a visual feedback to the user via the graphical interface:

```
<mmi:mmi xmlns:mmi="http://www.w3.org/2008/04/mmi-arch" version="1.0">
<mmi:doneNotification requestID="req1" source="MCRecognizer.php" target="myIM.php" context="myC" status="success">
      <mmi:data>
        <data id="detectionList">
          <users>
             <user id="u58" confidence=".85" />
             <user id="u32" confidence=".75" />
             <user id="u87" confidence=".60" />
          </users>
        </data>
      </mmi:data>
</mmi:doneNotification>
</mmi:mmi>
```

1.3.1.4 The MMI Framework & Architecture for Multimodal Systems

This section use the classification already presented[47] to analyze the W3C's proposal. As a generic framework and reference architecture, the MMI proposal is a resource containing a consistent set of architectural best practices to be used by different multimodal systems.

It is an architectural pattern -or a set of patterns-coming from a common agreement between international industrials and academic institutions. It can be partially or completely instantiated and it is in the process to be tested for use in the multimodal industry.

The MMI Framework & Architecture is an design pattern

For this reason, some of the characteristics of a *Multimodal System* are automatically covered by the MMI Framework & Architecture[48]. For example, by definition, this kind of architectural pattern must support multiple *modes*, *modalities* and *media* in a very generic and interoperable way **[Table 1.3.1]**.

47 See supra part 1.2. Definition of a Multimodal System

48 From now on, this paper will refer with the expression «MMI Framework & Architecture» to the MMI Runtime Framework Note and MMI Architecture and Interfaces candidate recommendation.

In contrast, we can point out some specific characteristics: the inputs and outputs are treated in an abstract way, and the architecture is fully bidirectional using *Sensor* and *Effector Systems* with dedicated functionalities to handle input/outputs **[Table 1.3.1]**. This is proposed from a multimodal perspective and takes into account the coordination (orchestration) of modalities. The *media* is described with the help of a complementary recommendations such as the EMMA markup language [W3C-EMMA 2009] that addresses the data source representation for input and output. This is completed also by the use of modality-focused languages such as VoiceXML [W3C-VoiceXML 2009], EmotionML [W3C-EmotionML 2011] or InkML [W3C-InkML 2011].

Finally, even if the specification does not give any details about the implementation of the *fusion* engines [Lalane et al. 2009], or the *fission* management [Rousseau et al. 2006], or the nature of the possible combinations with some *fusion / fission* criteria [Bouchet 2006]; the MMI Framework describes in a generic way the responsibilities of the Interaction Manager and the Modality Components concerning the *fusion* and *fission* tasks **[Table 1.3.1]**. Again, due to its generic nature, it allows the management of any kind of *fusion* or any *fission* mechanism.

Direction	**Both**		*YES*
Level of Abstraction	**Input Abstraction**	System	*INPUT MODALITY COMPONENT*
	Output Abstraction	System	*OUTPUT MODALITY COMPONENT*

Supported Modes	**Acoustic**	*YES*
	Visual	
	Haptic	
	Olfactive	
	Gustative	

Multiple Modalities for each Supported Mode ?	*YES*
Media Support	*YES*

Fusion	**Data**	*MODALITY COMPONENT OR INTERACTION MANAGER*
	Feature	
	Semantic	
	Hybrid	

Fission	**Data**	*MODALITY COMPONENT OR INTERACTION MANAGER*
	Modality	
	Semantic	

Table 1.3.1: Multimodal Characteristics in the MMI Framework & Architecture proposal.

To handle the behavior of the System the MMI Framework & Architecture focused on the Event Management. As we can see in **[Table 1.3.2]**, the main contribution of the MMI Framework and specially of the MMI Architecture, is oriented to the exchange of events between components.

This is followed by the generic definition of an interaction handler and a session handler, described as useful architectural modules but unfortunately for the moment with few details or guidelines about their behavior. The specification proposes three types of multimodality[49] sequential, simultaneous and composite and two interaction types: direct and free flow.

In addition, the session handling is for the moment described from the perspective of the context of interaction and not from a system's point of view. The uptime of components is implementation dependent, which means that the component's life-cycle, with discovery, registering, indexing, querying and removal is for the moment not explained. This issues are part of the current charter of the Working Group.

Finally, these specifications recommend the use of markup languages like SCXML for turn-taking or coordination of modalities but do not address the interfaces required for the decisional support or the context-awareness.

49 In contrast, for example with the CARE or the TYCOON approaches which are more detailed in the analysis of multimodal coordination types keeping a high level of abstraction.

As a result, we can affirm that according with the current workload in the Working Group, the main focus chosen by the MMI proposal for the management of the multimodal system behavior is the handling of the interaction Events through the Life-Cycle protocol proposal and a set of authoring guidelines.This characteristic allows a great expressiveness, but sometimes at the expense of completeness, which is normal for an international and multilateral specification that is a 'work in-progress'[50] looking for a more important interoperability between systems.

Session	Supported		*YES*			
	Handle by Manager		*YES*			
	Migrated		*NO*			
	Historized		*NO*			
Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*YES*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*YES*
	Disambiguated		*YES*	Associated to context		*NO*
	Multiple Granularity		*YES*			
Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous			Free flow	
		Composite				
	Synchronized		*YES*	**Focus Handled**		*YES*
	Historized		*NO*	**State recorded**		*NO*
	Turn Taking		*YES (SCXML)*	**Context updated**		*YES*
	Strategies	One	*YES*	**Interpretation**	by grammar	*YES*
					by structure	*YES*
		Multiple	*NO*		by planning	*NO*
					by learning	*NO*
					by intention int.	*NO*

Table 1.3.2: Multimodal Behavior in the MMI Framework & Architecture.

According with [Bouchet 2006] one of the criteria to analyze a *Multimodal System* is its support of constraints for human-computer interaction like the inclusion of some information about the user and the context in the description of the devices and the interaction. In other words, how the system handles what we called the Participants and the Usage of the Multimodal System.

Concerning the System's Participants, the MMI Framework & Architecture addresses the data storage of user and application information with the proposal of a Data Component. Nevertheless, at this moment the description of some generic interfaces or tasks of a such component is absent. Moreover, no guidelines are given to indicate the use of user models or user profiles.

On the other hand, the MMI Architecture specification provides a description of some basic development constraints in the lifecycle of multimodal applications, and manages multiple granularities in the model behind the mmi attributes and events **[Table 1.3.3]**. Yet, few information is given about Device modeling for multimodal purposes. On the other hand the Domain modeling of the interaction phenomenon is reduced, although this analysis is a basis to propose some generic mechanisms facilitating the coordination of multiple modalities. In contrast with this temporal lack of models for HCI support, it allows to define the participation of a device in multimodal sessions covering discovery, annotation or coordination of modalities.

To sum up, to support HCI constraints, the MMI Framework & Architecture proposes a Data Component and a description from which result some design rules. Actually, the next charter of the MMI Architecture addresses some of these issues, which is an opportunity to extend the recommendation and to cover the Device, User and Domain modeling in a basic way.

50 The MMI Architecture is currently a Candidate Recommendation of theW3C.

With respect to the management of the environment and context, the MMI Framework specification affirms that developers might be able to create applications that dynamically adapt to changes in device capabilities, user preferences and environmental conditions.

User Modeling	**At Model Level**		*NO*	**Data collection**		*NO*
	At Profile Level		*NO*	**Personalization**		*YES*
	Generation	Deducted	*NO*	**Levels**	Sensorial	*YES (EMMA)*
		Inferred	*NO*		Behavioral	*NO*
					Semantic	*NO*
	Stored	Centralized	*YES*	**Stereotypes used**		*NO*
		Distributed	*YES*	**Social support**		*NO*
Domain Modeling	**Application**		*YES (basic)*	**Interaction**		*YES (basic)*
	Presentation		*YES (markup proposed)*	**Use**		*NO*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*YES*	**Abstract**		*YES*
				Taxonomies used		*NO*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*NO*			

Table 1.3.3: Participants in the MMI Framework & Architecture proposal.

With this purpose, the MMI Framework must allow the Interaction Manager to determine what information is available. It suggest that environmental conditions can be monitored and reported to the Interaction Manager by inspecting interference conditions or events from environmental sources affecting the interaction. It also suggest that notifications can be used to indicate that the application should switch to an alternative mode of operation.

Concerning the temporal situation management, while the MMI Architecture privileged the treatment of temporal data as intervals, by choosing to address the continuos media processing control, the event-driven mechanism suggests that the exchanges are produced as a point series of commands. As a result, these discrete events are exchanged in a heterogeneous way depending on the interaction flow. In other words, while the interaction cycle and the analysis and interpretation of media are viewed as intervals distributed homogeneously, the user interaction and media commands are viewed as interruptions of this data flow, which are discrete events distributed heterogeneously.

The interaction context can be viewed as a *symbolic time* related to the multimodal interaction domain. This is an interval time composed by overlapped sequences that must be coordinated by the orchestration task in the Interaction Manager. For this task, the MMI Framework recommendation groups in two generic sets of sequential and parallel relations the 13 temporal relations of Allen[51]. Relational aspects are also covered by the specification in the MMI Interaction Events Life-Cycle protocol, for example on the sequential Request/Response pairs or the logical order of events in the interaction cycle. **[Table 1.3.4]**

Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*NO*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*YES*	**Temporal Operators**		*NO*

Table 1.3.4: Environment and context in the MMI Framework & Architecture.

51 See supra part 1.2.3.2 Temporal Situation

The MMI Framework foresees also a dedicated component, the System and Environment Component, but it is described mostly as a storage component for configuration data.

Nevertheless, neither the MMI Framework nor the MMI Architecture recommendation -in its current state-, give some details about the context or the environment viewed from the perspective of the usage or the spaces that could affect the multimodal interaction.

The Contribution of the MMI Framework & Architecture for a Multimodal Architecture of Reference

To sum up, the MMI Framework and the MMI Architecture proposals focus on a bidirectional, expressive and generic mechanism, very well detailed in subjects like communication protocol or event handling in multimodal systems. For this reason, and despite the temporary incomplete aspects of a work in progress, they are a rich starting point for the design of a multimodal application.

We have provided an in-depth analysis of the W3C's Open Standard approach based on the set of criteria given in section 1.2.

Outline

Our goal for the following section is to outline this aspects in other proposals of two kinds: the architectures generated by multimodal design frameworks and the empirical models coming from the study of a set of concrete implementations.
By analyzing each approach against these optional criteria, we will bring out the strengths of each approach against the goal of a Multimodal Architecture of Reference, with context-awareness in a large-scale network like the Internet. We suppose that by analyzing their distribution of features and responsibilities these could be two potential sources of generic components for such an architecture.

This comparison is based in a historical analysis of multimodal implementations and research trends[52] and it will serve as groundwork for the design of modality discovery and context management with semantic web technologies for the MMI Framework & Architecture.

1.3.2 Architectures from Multimodal Frameworks

The multimodal frameworks are indirectly a source of architecture models. Behind a framework we can find a proposal that models the kind of application that can be designed, what are the minimal components needed for certain responsibilities and what is their basic behavior. The next sections will cover some of these frameworks under the light of our analysis grid and from an architectural point of view.

Color codes to compare visually the studied architectures

For an easier comparison, the diagrams representing the studied architectures will resume the color codes already used, with fours additions.
- The handlers for input/output participants are represented with the color code **green**; the components responsible of interaction coordination are represented with the color code **orange**; the components responsible of data handling are represented with the color code **violet**; the components responsible of following the states of the system are represented with the color **turquoise**; and the component responsible to handle the application functions, represented with the color code **brown**.

In addition to these color codes, we propose:

- The component responsible to handle the decisional features, will be represented with the color code **pink**.
- The *fusion* engine will be represented in with the color code **blue** and a dotted line.
- The *fission* engine will be represented in with the color code **yellow** and a dotted line.
- The knowledge sources will be represented in with the color code **blue navy**.

52 See Appendix 1 Multimodal Architectures Timeline

1.3.2.1 THE OPEN AGENT ARCHITECTURE - 1994

The **OPEN AGENT ARCHITECTURE** is a research framework [Martin et al. 1994] for constructing agent-based systems, in which distributed collections of autonomous agents cooperate to provide software services.

Agent's definition in the Open Agent Architecture

In the **OPEN AGENT ARCHITECTURE** an Agent is defined as any software process that a) registers the service it can provide b) is able to use a predefined protocol of communication and c) shares functionalities common to all of these software process, such as the ability to install triggers, manage data in certain ways, etc. They are software components described using a human 'agent' metaphor.

The framework consists on the Facilitator Agent (in orange) and libraries in several languages, which can be used for constructing client agents of three types [**Figure 1.3.14**]:

- Application agents (in **brown**): to provide a collection of reusable services of a particular sort such as speech recognition, natural language processing, email, and some forms of data retrieval and data mining,

- Meta-agents (in pink): to assist the facilitator agent in coordinating the activities of other agents with rules, learning algorithms, planning,...

- User interface agents (in **green**): a collection of "micro-agents", each monitoring a different input modality -point-and-click, handwriting, pen gestures, speech- and collaborating to produce the best interpretation of the current inputs.

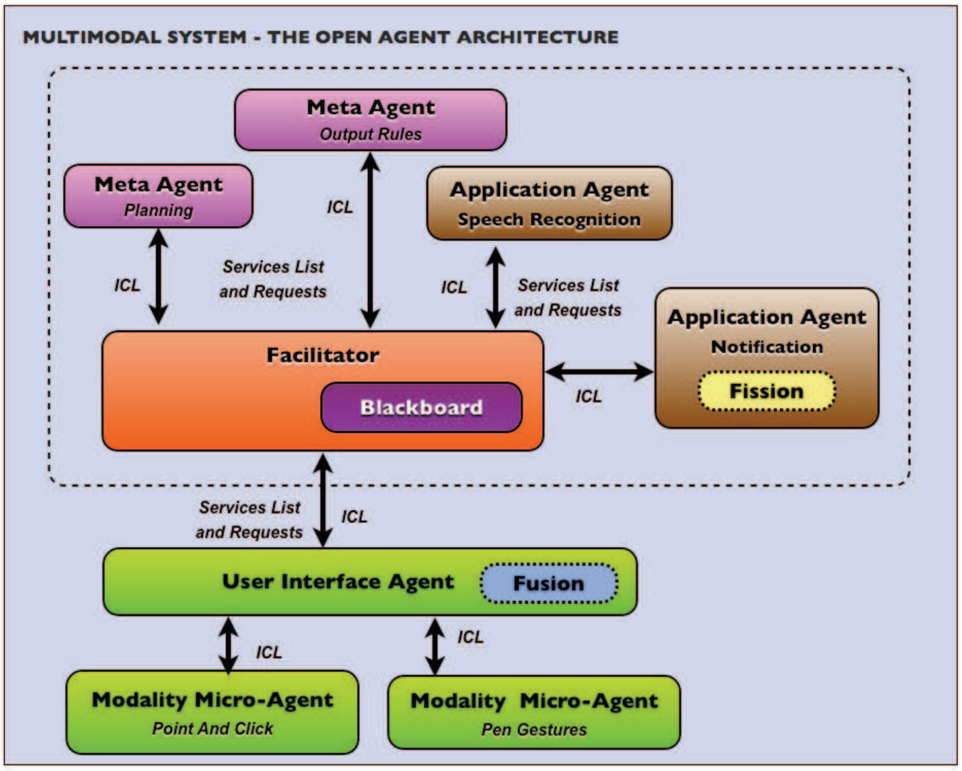


Figure 1.3.14: OAA's Multimodal Architecture

Each system includes one or more facilitators, each coordinating the communications and activities of a set of agents. The facilitator is itself a specialized server agent handling cooperative problem-solving.

The communication and cooperation between agents are dispatched by one or more facilitators, which are responsible for matching requests, from users and agents, with the descriptions of the capabilities of the possible target agents. However they are not centralized controllers, but rather coordinators, as they draw upon knowledge and advice from several different -and potentially distributed- sources to guide their delegation choices.

In some systems, the facilitator is also used to provide a global data store for its client agents, which allows them to adopt a blackboard style of interaction where a common knowledge base (in **violet**), the "blackboard" , is iteratively updated by knowledge sources. Each source can watch for items of interest, execute processes based on the state of the blackboard and then add partial results or queries that other processes can share.

In the **OPEN AGENT ARCHITECTURE** when invoked, a client agent makes a connection to a Facilitator, known as its parent facilitator, and informs it of the services it provides. Using the brokering services of its parent facilitator, a client agent can provide several types of services, including procedural goal fulfillment, maintenance and querying of data stores, and setting and responding to triggers of several types.

When a facilitator receives a request, it constructs a goal satisfaction plan and oversee its satisfaction in the most efficient manner consistent with the specified advice.

The declarations, requests, and maintenance commands associated with all of these types of services in the **OPEN AGENT ARCHITECTURE** are formulated and handled in a consistent, integrated fashion, using the Inter-agent Communication Language (ICL) [Finin et al. 1997]. ICL is the interface, communication, and task coordination language shared by all agents.

Requests for services provided by other agents are sent to the requester's parent facilitator, which, in turn, delegates them to one or more providers. Requests in the system does not require that the requester know about the targeted agent to handle the call, it is decided by the facilitator. Requests are annotated with parameters metadata, which serve as advice to the facilitator regarding the desired handling of the request.

Triggers provide a general mechanism to express conditional requests. Each agent can install triggers either locally, on itself, or remotely on its facilitator or peer agents. There are four types of triggers: communication triggers "fire" when a message matching a given pattern is sent or received; data triggers fire when a data store is updated in some specified manner; task triggers fire when some application-specific condition is met; and time triggers fire at a fixed time, or at fixed intervals.

The **OPEN AGENT ARCHITECTURE** provides an integral support for the construction of multimodal interfaces, including the handling of both directions (input/ output) in a level of abstraction higher than the device or the driver and with multiple modalities, modes and media (handled mostly by the application agents and the content layer of ICL) [**Table 1.3.5**].

Direction	**Both**		*YES*
Level of Abstraction	**Input Abstraction**	System	*AGENT*
	Output Abstraction	System	*AGENT*
Supported Modes	Acoustic		*YES*
	Visual		*YES*
	Haptic		*YES*
	Others		*NO*
Multiple Modalities for each Supported Mode ?			*YES*
Media Support			*YES*
Fusion	Data		*IN AGENTS*
	Feature		*IN AGENTS*
	Semantic		*IN AGENTS*
	Hybrid		*IN AGENTS*
Fission	Data		*IN AGENTS*
	Modality		*IN AGENTS*
	Semantic		*IN AGENTS*

Table 1.3.5: Multimodal Characteristics in OAA.

It supports also simultaneous work over shared data and processing resources between users and agents by distributing the *fission* and *fusion* processes as dedicated responsibilities of the User Interface Agents or Application Agents. [**Figure 1.3.5**]

The **OPEN AGENT ARCHITECTURE** is also flexible enough to assemble service providers—both at development time and at runtime, it allows to add new agents handling new types of modalities and media. [**Table 1.3.5**]

In the studied papers describing the **OPEN AGENT ARCHITECTURE** a description of a multimodal session is absent, while the metaphor of a conversation is used to focus on the cycles of user-system interaction leaving aside the multimodal system global state during interaction. This metaphor leads mainly to a conversational protocol layer for the Inter-agent Communication Language (ICL) defined by some Event types and their attribute list. These characteristics are detailed in the Session and Event lines of [**Table 1.3.6**]:

Session	Supported		*YES*			
	Handle by Manager		*NO*			
	Migrated		*NO*			
	Historized		*NO*			
Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*YES*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*YES*
	Disambiguated		*YES*	Associated to context		*YES*
	Multiple Granularity		*YES*			
Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous	*YES*		Free flow	
		Composite	*NO*			
	Synchronized		*YES*	**Focus Handled**		*YES*
	Historized		*YES*	**State recorded**		*YES*
	Turn Taking		*YES*	**Context updated**		*NO*
	Strategies	One	*NO*	**Interpretation**	by grammar	*NO*
					by structure	*YES*
		Multiple	*YES*		by planning	*YES*
					by learning	*YES*
					by intention int.	*NO*
Decision	**Goal**	Satisfy constraints	*YES*	**Techniques**	Formal	*NO*
		Reduce options	*YES*		Non Formal	*YES*
		Recommend	*YES*			
	Knowledge	System	*YES*	**Uses**	Resolve ambiguity	*YES*
					Uncertain reasoning	*NO*
		Domain	*YES*		Support interpretation	*NO*
					Collaborate in planning	*YES*
		General	*NO*		Support RW understanding	*NO*

Table 1.3.6: Management of Multimodal Behavior in the OAA.

Concerning the Interaction the Facilitator plays the role of an assistant for requesters and providers for making contact. It ensures a transparent delegation with its loose-coupled solution: requester agents don't need any knowledge of the identities, the number or the locations of provider agents. It handles the strategies for optimization and the parallelism of requests, the coordination of request to the satisfying agents (the *fusion)* and the assembling of their responses into a coherent whole to return (the *fission)*. And the Facilitator can employ strategies and advices given by the requesting agent to satisfy the request. [**Table 1.3.6**]

The Decision management is focused on problem solving with the procedure solvables and the data solvables in Agents. A procedure solvable performs an action whereas a data solvable provides access to a collection of data[53]. One of the parts of a solvable is a goal, which is a logical representation of the services that the solvable provides. An Agent request services by delegating goals to its Facilitator by giving constraints, by reducing options and by giving recommendations. [**Table 1.3.6**] For example, helped with advice parameters of high-level like the «solution_limit» the facilitator constructs a goal satisfaction plan consistent with the specified advise. In short, the Facilitator can employ strategies and advices given by the requester through the request parameters.

53 The detailed description of solvables is out of the scope of this analysis. For more information see [Martin et al. 1994]

For the Device Modeling the architecture proposes the advertisement of device capabilities as a list of services, the solving API's proposed by the Facilitator and the suggestion of data solvables in client Agents. [**Table** 1.3.7]

One of the goals of the architecture is to treat users as privileged participants of the system. Nevertheless, no information is given about how the user information is modeled or handled in the system or which interfaces are available for the personalization of applications. It seems that some preference-based mechanisms are foreseen at sensorial level (how to display, the sound volume) but this is only suggested by some examples.

On the other hand, domain modeling is focused in the life-cycle process with some requirements like to minimize the effort required to create new agents, to wrap existing applications or to encourage reuse of both domain-independent and domain-specific components.

With this goal, for us one of the most important proposals are the Meta-Agent modules, which are domain-specific sources of information to complete the generic description of the domain. The architecture also proposes task triggers to support domain -dependent behaviors.

Device Modeling	**At Model Level**		*NO*	**Abstract Description**		*NO*
	At Profile Level		*NO*	**Multimodal Info**		*YES*
	Managed	by Document	*YES*			
		by API	*YES*			
		by Service	*YES*			
		by Repository	*YES*			
User Modeling	**At Model Level**		*NO*	**Data collection**		*NO*
	At Profile Level		*NO*	**Personalization**		*YES (not described)*
	Generation	Deducted	*NO*	**Levels**	Sensorial	*YES*
		Inferred	*NO*		Behavioral	*NO*
					Semantic	*NO*
	Stored	Centralized	*NO*	**Stereotypes**		*NO*
		Distributed	*YES*	**Social support**		*NO*
Domain Modeling	**Application**		*YES - AGENTS*	**Interaction**		*YES (blackboard)*
	Presentation		*NO*	**Use**		*NO*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*NO*	**Abstract**		*NO*
				Taxonomies		*NO*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*NO*			

Table 1.3.7: Participants in the OAA.

To handle the interaction situation, the **OPEN AGENT ARCHITECTURE** do not covers the usage situation or the spatio-temporal information. The temporal situation is handled [**Table** 1.3.8] explicitly with the time triggers mechanism and the sequential and parallel support of requests in the Facilitator. The application agents offer a flexible support to handle relational aspects in the management of the time.

Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*NO*
	Sensed Data Distribution	Homogeneous	*NO*	**Returned Data Distribution**	Homogeneous	*NO*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*YES*	**Temporal Operators**		*NO*

Table 1.3.8: Environment and context in the OAA.

The Contribution of the Open Agent Architecture for a Multimodal Architecture of Reference

To sum up, we can point out that the main contribution of **OPEN AGENT ARCHITECTURE** from the perspective of a Multimodal Architecture of Reference is the proposal of the delegation mechanism based on the service advertisement and metadata exchanges. Input and output process are modeled in generic and reusable structures. The mechanism of triggers, their classification and the remote installation of triggers is also an interesting contribution. The architecture is extensible with the Meta Agents and the Applications Agents offering a mechanism to support the application-specific needs by keeping a high level perspective.

1.3.2.2 CICERO - 1995

The attempt to tackle the problem of information overload in multimodal applications appears in 1995 with **CICERO** [Arens et al. 1995], a context-aware system [**Figure 1.3.15**] proposed to dynamically allocate the information needed to a presentation. This is a generic interaction platform that can be tailored as needed to be reused in other environments.

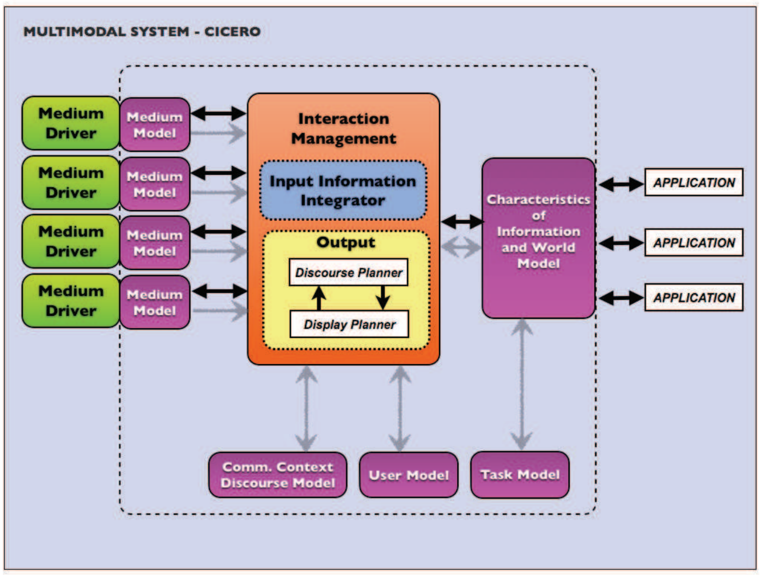


Figure 1.3.15: CICERO's Multimodal Architecture

In **CICERO**, the Interaction Manager (in **orange**) has two responsibilities: to handle the domain and the discourse data and to dynamically coordinate the operation of the media and information resources available.

On the input side, two modules are used: the input media themselves -in **green**

(with the associated models-in **violet**) and the fusion engine which is called Input Information Integrator module (in **blue** -dotted).

The input modules depend also on a set of generic and specific semantic models that converts the concrete input into a Virtual Device that is a model of a particular I/O functionality that may be realized by numerous different combinations of hardware and/ or software. Any actual device is a concrete instantiation of the Virtual Device.

The capabilities of each medium and the nature of the contents of each information source are semantically described in a single uniform knowledge model (in **violet**). Each Virtual Device should contain, in addition to parameters for functionality, specifications and evaluation measures for at least the following:

- Device: resolution, latency, tolerances, data accuracy, data rate, etc.
- User: (cognitive) short-term memory requirements, communication protocol -language, icons, special terminology-, etc. (sensorial) muscle groups and actions required, associated fatigue factors, etc.
- Usage: techniques of use -traditional menu vs. pie-chart menu-, performance characteristics, etc.

On the output side, the fission engine is treated as a Presentation Planner (in yellow -dotted) with two linked reactive planners that perform the construction of the discourse on runtime and the allocation planning of media.

To compose a presentation the intelligent manager coordinates and synchronizes the various media in a way that decreases the system complexity caused by information overload. This is made by building an interface that designs itself at run-time and adapts to the changing demands of information presentation.

The nature of the contents of each information sources to compose are semantically modeled in **CICERO** as abstract information types: these are the characteristics of the information to be displayed, the application task, the discourse and communicative context and the participants' goals, interests, abilities and preferences.

Based on these planning decisions **CICERO** can dynamically compose an interface at run- time.

Direction	**Output**	*YES (presentation-oriented)*	
Level of Abstraction	**Input Abstraction**	System	*VIRTUAL DEVICE*
	Output Abstraction	System	*VIRTUAL DEVICE*

Supported Modes	Acoustic	*YES*	**Multiple Modalities for each Supported Mode ?**	*YES*
	Visual			
	Haptic	*NO*		
	Others	*NO*	**Media Support**	*YES*

Fusion	Data	*NO*	**Fission**	Data	*INPUT INFORMATION INTEGRATION*
	Feature	*PRESENTATION PLANNER*		Modality	
	Semantic			Semantic	
	Hybrid	*NO*			

Table 1.3.9: Multimodal Characteristics in CICERO.

With the inputs and outputs handled as Virtual Devices **CICERO** embraces an approach with an abstraction that views both at a system level **[Table 1.3.9]**.

In the date of the proposal, only two modes were addressed, due to the technical difficulties on handling the other modes at that time.

The fusion engine is an intelligent planning component composing the presentation on the basis of metadata information while the fission engine is not described with the same level of detail.

Interaction (Dialog)	Multimodality	Sequential	*NO*	Interaction Types	Direct	*YES*
		Simultaneous			Free flow	
		Composite				
	Synchronized		*NO*	Focus Handled		*YES*
	Historized		*NO*	State recorded		*NO*
	Turn Taking		*NO*	Context updated		*NO*
	Strategies	One	*YES*	Interpretation	by grammar	*YES*
					by structure	*YES*
					by planning	*YES*
		Multiple	*NO*		by learning	*NO*
					by intention int.	*NO*
Decision	Goal	Satisfy constraints	*YES*	Techniques	Formal	*NO*
		Reduce options	*YES*		Non Formal	*YES*
		Recommend	*NO*			
	Knowledge	System	*YES*	Uses	Resolve ambiguity	*YES*
					Uncertain reasoning	*NO*
		Domain			Support interpretation	*YES*
					Collaborate in planning	*YES*
		General			Support RW understanding	*NO*

Table 1.3.10: Management of Multimodal Behavior in CICERO.

In **CICERO** the multimodal session is not addressed, except by the suggestion of a discourse structure, a tree-like structure that represents the organization and contents of the interaction at the current time. This presentation plan is asymmetric, which means that no information is given about the effects and links between the input/output media.

The Planning approach of Cicero

CICERO is not described as an event-driven system. This means that the two central processing modules are described as planners: one to plan the underlying discourse structure the other to allocate media. A reactive planner model is proposed but no information is given about the event management performed by this planner.

This planning approach is oriented to natural language processing, focused on the discourse management and the presentation of information. By its nature it does not give any suggestion about multimodal handling or some specific turn taking mechanisms. It rather focuses on the dialog interpretation and presentation based on structures and planning strategies. **[Table 1.3.10]**

The other focus of this proposal is the knowledge handling needed by the planning tasks and the interpretation of the models of the world surrounding the interaction. **[Table 1.3.10]** For example, in the content planning process operators construct the discourse structure taking decisions based on goals, data characteristics and user knowledge and interests.

On the other hand, the media display process decides which presentation to compose based on the presentation history, data characteristics and current state. In this way, the decision mechanism resolves ambiguity, supports interpretation and collaborates in planning **[Table 1.3.10]**

Concerning the System's Participants, **CICERO** does not describe any data storage or the data handling with a dedicated component. Nevertheless, detailed guidelines are given for device modeling, user modeling and domain modeling [**Table 1.3.11**].

The models are described in documents written in the Loom representation language [MacGregor 1988]. Models are not inferred because there is no mechanism of data collection.

The user model and the domain model are based on categories but the user model does not use stereotypes or on some social information and the domain model lacks of theoretical foundation coming from theoretic tools like the interaction design spaces or the analysis of temporal relations between modalities.

Device Modeling	**At Model Level**		*YES*	**Abstract Description**		*YES*
	At Profile Level		*YES*	**Multimodal Info**		*YES*
	Managed	by Document	*YES*			
		by API	*NO*			
		by Service	*NO*			
		by Repository	*NO*			
User Modeling	**At Model Level**		*YES*	**Data collection**		*NO*
	At Profile Level		*YES*	**Personalization**		*YES*
	Generation	Deducted	*YES*	**Levels**	Sensorial	*YES*
		Inferred	*NO*		Behavioral	*YES*
					Semantic	*YES*
	Stored	Centralized	*YES*	**Stereotypes**		*NO*
		Distributed	*NO*	**Social support**		*NO*
Domain Modeling	**Application**		*YES (basic)*	**Interaction**		*YES (basic)*
	Presentation		*YES*	**Use**		*YES*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*YES*
	Interaction Classification	Categories	*YES*	**Abstract**		*YES*
				Taxonomies		*YES*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*NO*			

Table 1.3.11: Participants in CICERO.

By focusing on the information overload and the context-awareness, **CICERO** mainly contributes with its modeling effort to the treatment of the usage situation, the temporal situation and the spatio-temporal situation [**Table 1.3.12**].

Semantic Models in Cicero

Five semantic models are produced, based on the collection of a basic thesaurus that could be needed to describe the minimal configuration of a dynamic human-computer interface.

The first model, «the model of media characteristics» is the model that covers the requirements for device modeling [**Table 1.3.6**] since it describes the device's parameters relevant to every class of generic device (hardware, software, cognitive requirements, physical requirements, mode of use and performance) and the media attributes for each class (medium, exhibit, substrate, info. carrier, carried item, channel, internal semantic system...).

The second model, «the model of characteristics of information to be displayed or input» describes the intrinsic properties of every item on the layout produced by the fission mechanism, properties associated to its class and properties of the chosen layout. This model covers the requirements for domain modeling [**Table 1.3.12**] (for presentation, taxonomies, interaction and application information).

The requirements for usage situation modeling [**Table 1.3.12**] are covered in a fourth «model of application tasks and interlocutors' goals». This model describes the mental models and knowledge states (including emotion) involved in the interaction.

In this way, in the «model of application tasks and interlocutors' goals» as in the «model of discourse and communicative context» the usage situation modeling, the temporal situation modeling and the space-time situation [**Table 1.3.12**] are relatively covered with the study of the temporal, spatial and social relationships between interlocutors.

Finally the fifth «model of user's goals, interests, abilities and preferences» covers some of the requirements for User modeling showed in [**Table 1.3.11**]. This user model describes the characteristics of the user (presentation display preferences, knowledge of the topic, language ability, interest in the topic, opinions of the topic) but lacks of a social perspective based on stereotypes and does not propose attributes for collecting multimodal data about the user.

Usage Situation	**At Model Level**		*YES*	**Relationships**		*YES*
	At Profile Level		*YES*	**Mental Models**		*YES*
	Factual Description		*NO*	**Social Qualities**		*NO*
	Interpretation Description		*YES*			
Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*YES*	**Temporal Operators**		*YES*
Space-Time Situation	**Ontologies**		*NO*	**Emotional Bias**		*YES*
	Other Context Descriptions		*YES*	**Social Bias**		*NO*
	Coordinates	Geometric	*NO*	**Proxemic**	Distance	*YES*
		Symbolic	*NO*		Dynamics	*YES*
	Relations	Metrical	*NO*	**Qualitative Relations**	Quantity Spaces	*YES*
		Ordinal	*NO*		Cardinal	*YES*
		Topological	*YES*		Grid-Based	*NO*
	Views	Allocentric	*YES*	**Naïve Operators**		*NO*
		Egocentric	*YES*			

Table 1.3.12: Environment and context in CICERO.

The Contribution of Cicero for a Multimodal Architecture of Reference

To sum up, we can point out that the main contribution of CICERO from the perspective of a Multimodal Architecture of Reference is the development of multimodal generic models. These generic models are proposed in two forms, the Virtual Device abstraction and five models of generic requirements (which can be used as taxonomies) to be described by a multimodal system.

1.3.2.3 GALATEA - 2003

GALATEA [Nitta et al. 2003] is a toolkit developed by 16 research institutes in Japan. Its implementation includes a Japanese speech recognition engine, a Japanese speech synthesis engine, and a facial image synthesis engine, a dialogue manager that can integrates multiple modalities, interprets them, and decides an action with differentiating it to multiple media of voice and facial expression.

Using the **GALATEA** toolkit designers can easily develop their unique life-like agents that communicate with users via spoken language.

It is a Model-View-Controller architecture, which separates the application logic to the user interface description intermediated by the controller.

The proposed architecture a) aggregates the processing depending on the modalities with the proposal of a user interface description b) facilitates the addition of new modalities with the separation of this user interface description, c) and enables the timing-sensitive modality control with the event-driven facet of the MVC model.

New modalities can be implemented into **GALATEA** through the authoring framework which is a prototyping tool for coding the interaction flow as an scenario document stored in a database.

Most of the software components are modularized and an Interaction Management Module composed of an Agent Manager (in orange) and a Task/Dialog Manager (in brown) can control these components under distributed environments [**Figure 1.3.16**].

The Agent Manager works as a communication hub -inspired on theGalaxy II system [Seneff et al. 1998] for input/output functional modules, through which they communicate each other.

These components are modularized independently, and an input/output device is directly controlled by its related module. If a new module is added to the system, it can be done connecting the module to the Agent Manager.

Thus, the Agent Manager serves as an integrator of all the modules of the multimodal system and as a communication dispatcher.

The Agent Manager is a *fission* engine (yellow -dotted), because it synchronizes speech synthesis and facial image animation to achieve the precise lip-sync. And the Agent Manager works also as a *fusion* engine (**blue** -dotted) because it interprets and process the macro-commands coming from the Task Manager by expanding each received macro-command in a sequence of commands and sending them sequentially to the designated modules.

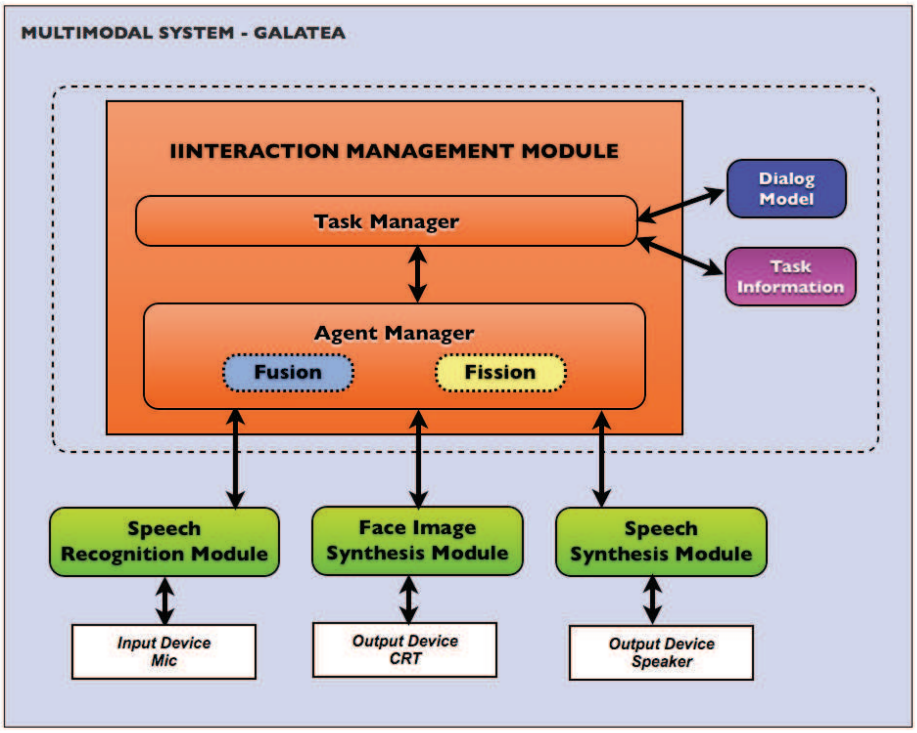


Figure 1.3.16: GALATEA's Multimodal Architecture

The Task/Dialog Manager communicates with the Agent Manager to achieve the interaction tasks described on a dialog scenario.

This is a turn taking controller based on task descriptions for designing and analyzing the human-machine dialogues. These tasks are described with two complementary languages.

The Primitive Dialogue Operation Commands (PDOC) plays the role of low-level language that is close to the device events and sequence control, while VoiceXML [W3C-VoiceXML 2003] plays the role of the high-level language that handles the task- oriented information and the intentions of the participants.

The **GALATEA** toolkit is limited to a few modalities and is very oriented to the recognition task. With two roles, the Agent Manager is mostly responsible of the communication tasks and few information is given about the *fusion* and *fission* processes that seems to be handled at the feature or at the modality level [**Table 1.3.13**].

Direction	Both	*YES*	
Level of Abstraction	Input Abstraction	System	*MODULE*
	Output Abstraction	System	*MODULE*
Supported Modes	Acoustic	*YES*	
	Visual		
	Haptic	*NO*	
	Others	*NO*	
Multiple Modalities for each Supported Mode ?	*NO*		
Media Support	*NO*		
Fusion	Data	*NO*	
	Feature	*YES*	
	Semantic	*NO*	
	Hybrid	*NO*	
Fission	Data	*NO*	
	Modality	*YES*	
	Semantic	*NO*	

Table 1.3.13: Multimodal Characteristics in GALATEA.

In GALATEA the Agent Manager serves as an integrator but no information is given about the way this integrator handles the multimodal session. The Agent Manager possesses a macro-command interpreter to expand each received macro-command in a sequence of commands [**Table 1.3.14**], but there is no description about how multimodal events are treated or exchanged by these modules.

Thus, the interaction is mostly sequential. On the other hand, with the use of VoiceXML the Task Manager can cover two types of dialogues, direct and free flow. GALATEA also proposes a synchronized interaction, with a history and state recorded to update context and a turn taking mechanism.

Dialog is handled by multiple strategies and the described solutions for interpretation are very complete: they only lack of a learning facet [**Table 1.3.14**]

In contrast decision making is addressed only for one modality: the speech synthesis and lip-sync module that uses Hidden Markov Models. It is used to resolve ambiguity and collaborate in planning at a system level. [**Table 1.3.14**]

Interaction (Dialog)	Multimodality	Sequential	*YES*	Interaction Types	Direct	*YES*
		Simultaneous	*NO*		Free flow	
		Composite	*NO*			
	Synchronized		*YES*	Focus Handled		*YES*
	Historized		*YES*	State recorded		*YES*
	Turn Taking		*YES*	Context updated		*YES*
	Strategies	One	*NO*	Interpretation	by grammar	*YES*
		Multiple	*YES*		by structure	*YES*
					by planning	*YES*
					by learning	*NO*
					by intention int.	*YES*
Decision	Goal	Satisfy constraints	*YES*	Techniques	Formal	*NO*
		Reduce options			Non Formal	*YES*
		Recommend				
	Knowledge	System	*YES*	Uses	Resolve ambiguity	*YES*
		Domain	*NO*		Uncertain reasoning	*NO*
		General	*NO*		Support interpretation	*NO*
					Collaborate in planning	*YES*
					Support RW understanding	*NO*

Table 1.3.14: Management of Multimodal Behavior in GALATEA.

GALATEA handles input/output process with a driver abstraction. Every modality is controlled by its related module but there is a lack of information about how this relationship is constructed or which kind of models or profiles are used. [**Table 1.3.15**]. Interaction Modeling inherits from the models behind the chosen languages to manage tasks.

User modeling is absent while domain modeling is mostly oriented to the authoring process. The design task is described with a semantic granularity coming from spoken dialog systems.

Hence, as a dialog system the architecture is not intended to be extensible to other modalities.

Device Modeling	At Model Level		*NO*	Abstract Description		*YES*
	At Profile Level		*NO*	Multimodal Info		*YES*
	Managed	by Document	*YES*			
		by API	*NO*			
		by Service	*NO*			
		by Repository	*NO*			
Domain Modeling	Application		*NO*	Interaction		*YES*
	Presentation		*YES*	Use		*NO*
	Life-cycle	Design	*YES*	Granularity	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*YES*
	Interaction Classification	Categories	*NO*	Abstract		*NO*
				Taxonomies		*NO*
		Design Spaces	*NO*	Performance Semantics		*NO*
		Temp. Relations	*NO*			

Table 1.3.15: Participants in GALATEA.

The temporal or the spatial situation are not handled and no contextual information is given, excepting the mental models needed to interpret and generate the human speech.

Social Information in Galatea

Finally, **GALATEA** toolkit handles social information coming from dialect recognition and in this concrete case, the cultural and semantic differences between the Kanji and the Kana pronunciation.

Usage Situation	At Model Level	*YES*	Relationships	*NO*
	At Profile Level	*YES*	Mental Models	*YES*
	Factual Description	*NO*	Social Qualities	*YES*
	Interpretation Description	*YES*		

Table 1.3.16: Environment and context in GALATEA.

The Contribution of Galatea for a Multimodal Architecture of Reference

To sum up, we can point out that the main contribution of **GALATEA** from the perspective of a Multimodal Architecture of Reference is the processing of macro-commands coming from the Task Manager by expanding or processing them.

Generic macro-commands are tools for the management of information and events at a higher abstract level which can ensure the architecture extensibility.

On the other hand, the use of two complementary granularities (at a low-level close to the device events and at a high-level close to the user goals) for the management of the interaction tasks is also an important contribution to the discussion around a generic multimodal architecture.

1.3.2.4 ICARE - 2004

ICARE [Bouchet et al. 2004] is a component-based platform for building multimodal applications focused on input processes. Based on the CARE conceptual model[54], this framework includes elementary components and composition components to combine modalities.

The CARE conceptual Model

Elementary components are divided on two types: the Device component and the Interaction Language component [**Figure 1.3.17**]. These are basic components that allow the designer to use «pure» modalities defined as the association of a physical device with an representational system (also called Interaction Language):

modality ::= 〈 *p* , *r* 〉 I 〈 modality , *r* 〉

where
p is a physical device
r is an Interaction Language (also known as a representational system:
a conventional set of signs to represent information)

Definition 1.3.1: A modality according to [Nigay et al. 1996] .

A Device component (in **green**) represents the physical input or output of the system. This is the lowest abstraction level in a given modality. A Device captures or renders the information in a system.

The Interaction Language component (in **green**) defines a set of well-formed expressions that are meaningful to the *multimodal system*. It is the logical level of a modality with a data structure defined specifically for the system, in other words, it is the set of metadata used by the system to represent the information produced by the interaction or for the interaction.

The Interaction Language component represents an abstraction layer for the physical object represented by the Device Component in addition to the peripheral driver. This is the layer covering the representational system [Bersen 1993].

This abstract layer enriches the raw data coming from the Device Component with a set of predefined metadata about:

- the sensorial conditions of the interaction (bootstrapping, timestamp),
- the usage (manipulability, passive or active modality)
- the domain-specific information (confidence factors of the captured raw data)
- the spatio-temporal conditions (the context of interaction).

According to this conceptual model, some examples of Modalities could be < keyboard,natural language >, < mouse , graphical form >, < microphone, natural language >, < smartphone, gesture language >.

On the other hand, the composition components are based on the CARE properties. [Nigay et al. 1997] As we describe before, these properties represent the relations of Complementarity, Assignment, Redundancy, and Equivalence that may occur between the modalities in a multimodal user interface. These are four internal relations between Device components and Interaction Languages or external relations between Interaction Languages and tasks handled by the Dialog Manager (in **orange**).

Some Composition components (in **orange**) are defined in **ICARE** to combine data from 2 to N components: one for Complementarity, one for Redundancy, and one for an alternative to Redundancy / Equivalence. These Composition components also enrich the data by adding the same type of metadata already described (time stamp, confidence factor, etc..) and by applying a *fusion* mechanism.

54 See supra part 1.2.2.3 Domain Modeling (the interaction Model).

In this way, Composition components are generic. They are not dependent on a particular modality and they enable the parallel (or sequential) use of multiple modalities combined in an abstract manner. **ICARE** defines a new component model based on Java Beans, and requires the components to be written in Java.

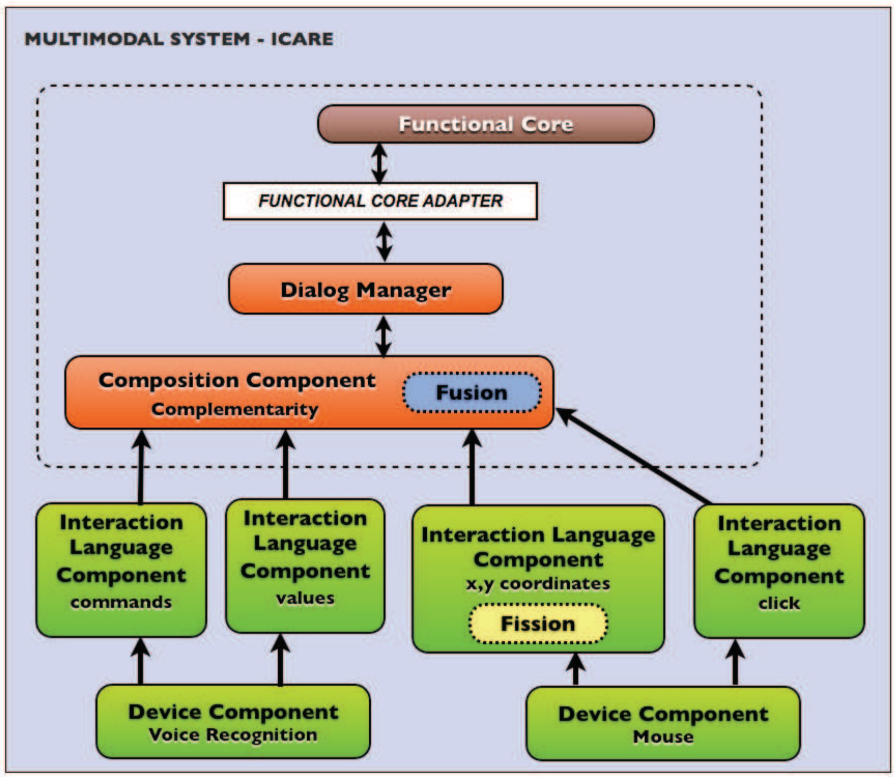


Figure 1.3.17: ICARE Multimodal Architecture

ICARE focuses on the input side of human-computer interaction [**Table 1.3.17**]. The input interaction processes are modeled at three levels of abstraction: the physical level in Device components, the representational level with the Interaction language components and the coordination level with the Composition components. This model also addresses the media support.

Semantic fusion, *feature fusion* and *data fusion* are handled by the Composition components and even some *fission* can be executed with the raw input data. The Composition components are generic modules dedicated to handle the coordination. They are based on the strategies of interaction.

Direction	Input	*YES*	
Level of Abstraction	Input Abstraction	System	*YES*
	Output Abstraction	*NO*	

Supported Modes	Acoustic	*YES*	Multiple Modalities for each Supported Mode ?	*YES*
	Visual			
	Haptic			
	Other	*NO*	Media Support	*YES*

Fusion	Data	*YES*	Fission	Data	*YES*
	Feature			Modality	*NO*
	Semantic			Semantic	*NO*
	Hybrid				

Table 1.3.17: Multimodal Characteristics in ICARE.

Concerning the behavior of the system [**Table 1.3.18**], the **ICARE** proposal does not propose a feature corresponding to our adopted definition of a multimodal session. In contrast, it proposes a coherent structure for communicating events: the ICAREEvent. This event structure is a set of attributes defined to facilitate the mechanisms with a strong temporal aspect. This attributes are : data, trust factor, initial timestamp, current timestamp, average initial time and average current time. By its generality, this event can transport monomodal and multimodal data in a single granularity but a protocol is not proposed.

Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*NO*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*YES*
	Disambiguated		*NO*	Associated to context		*NO*
	Multiple Granularity		*NO*			
Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous			Free flow	
		Composite				
	Synchronized		*YES*	**Focus Handled**		*YES*
	Historized		*NO*	**State recorded**		*YES*
	Turn Taking		*YES*	**Context updated**		*YES*
	Strategies	One	*NO*	**Interpretation**	by grammar	*NO*
					by structure	*YES*
		Multiple	*YES*		by planning	*YES*
					by learning	*NO*
					by intention int.	*NO*
Decision	**Goal**	Satisfy constraints	*YES*	**Techniques**	Formal	*YES*
		Reduce options			Non Formal	*NO*
		Recommend				
	Knowledge	System	*YES*	**Uses**	Resolve ambiguity	*NO*
					Uncertain reasoning	*NO*
		Domain	*YES*		Support interpretation	*YES*
					Collaborate in planning	*YES*
		General	*NO*		Support RW understanding	*NO*

Table 1.3.18: Management of Multimodal Behavior in ICARE.

Concerning the Interaction, **ICARE** covers the management of multimodality with a very complete set of properties going beyond of our sequential, simultaneous and composite criteria. In a certain amount, this approach can handle free flow interaction with the Composition Components supporting multiple strategies. This allows the interpretation of the inputs based on structures and planning. In contrast, the learning aspects or the intention interpretation are not described. [**Table 1.3.18**]

Concerning the Decision, **ICARE** uses a set of formal rules in the combination mechanism. As a design framework, the use of rules is not oriented to resolve ambiguity, uncertain reasoning or to support real world understanding. In contrast, these rules support the input interpretation and collaborate on planning at the design of the system. [**Table 1.3.18**]

Component's Metadata

The metadata model that is used to define the properties of the Device components and the Composition components covers [**Table 1.3.19**]:

- the sensorial modes with the properties addressing the communication mode with metadata like «gestural» to annotate the mouse or the keyboard, «oral» to annotate a microphone, or «mental» to annotate the brain activity [Bellik 1995].
- expertise level with properties to annotate the expertise required to use the device with metadata like «expert», «intermediate» or «novice».
- multimodal information with properties to annotate the combination nature with metadata related to multimodal design spaces.
- and trust level with values between 1 and 100 representing the pertinence an the level of trust of the generated information.

A User Model is advised but not detailed. In contrast, a User Profile is drafted in the metadata properties of the Device component (sensorial modes, behavioral expertise level). This profile is distributed in components, not collected and supports personalization. [**Table 1.3.19**]

Domain modeling in **ICARE** takes advantage of the CARE properties to classify the interaction phenomenon. This is completed by the use of taxonomies, temporal relations,and categories. In this way, the *fusion* of modalities is founded on theoretical studies about human-computer multimodal interaction.

Device Modeling	**At Model Level**		*YES*	**Abstract Description**		*YES*
	At Profile Level		*YES*	**Multimodal Info**		*YES*
	Managed	by Document	*YES*			
		by API	*NO*			
		by Service	*NO*			
		by Repository	*NO*			
User Modeling	**At Model Level**		*NO*	**Data collection**		*NO*
	At Profile Level		*YES*	**Personalization**		*YES*
	Generation	Deducted	*YES*	**Levels**	Sensorial	*YES*
		Inferred	*NO*		Behavioral	*YES*
					Semantic	*NO*
	Stored	Centralized	*NO*	**Stereotypes**		*NO*
		Distributed	*YES*	**Social support**		*NO*
Domain Modeling	**Application**		*YES*	**Interaction**		*YES*
	Presentation		*NO*	**Use**		*YES*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*YES*	**Abstract**		*YES*
				Taxonomies		*YES*
		Design Spaces	*YES*	**Performance Semantics**		*NO*
		Temp. Relations	*YES*			

Table 1.3.19: Participants in ICARE.

The metadata model used to define the properties of the components reflects also the Usage situation and the Spatio-temporal situation by the description of the proxemics property «place of nominal interaction» from an allocentric perspective[55] or the property «number of spatial dimensions». The first is the geographical place where the user must focus its attention to produce data in normal conditions, while the second is the spatial dimensions represented in a representational system (e.g. the Interaction Language of a Modality). [**Table 1.3.20**]

Mental models and relationships are reflected by the analogical and arbitrary properties of the Interaction Language metadata. Both properties cover the resemblance to reality of the representation (e.g. the use of coding or metaphors) and the heuristics in the definition of modalities (e.g. to use the space bar to brake in a game).

Other properties like the temporal dimension, the average processing time, the data transmission frequency, or the combination nature of the Composition components are some examples of the support of the Temporal situation on the **ICARE** platform. [**Table 1.3.20**]

55 A spatial reference frame based on the external environment and independent of one's current location on it. See supra part 1.2.3.3 Space-time situation (the proxemics interpersonal distances).

To sum up, the **ICARE** platform is a very complete reference with the use of design spaces to explain and design the interaction cycles in multimodal systems. The Device model is very detailed and the solution based on generic components to handle the multimodal composition allows the architecture extensibility and expressivity. The metadata model that describes the component's properties addresses contextual and device issues from the perspective of the fusion strategies at multiple levels. All these properties can enrich a reference architecture proposal from multiple perspectives.

Unfortunately, the platform only covers input processes without giving any suggestions about how this input proposal could interface with the output side.

Usage Situation	**At Model Level**		*YES*	**Relationships**		*YES*
	At Profile Level		*YES*	**Mental Models**		*YES*
	Factual Description		*NO*	**Social Qualities**		*NO*
	Interpretation Description		*NO*			
Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*YES*	**Temporal Operators**		*YES*
Space-Time Situation	**Ontologies**		*NO*	**Emotional Bias**		*NO*
	Other Context Descriptions		*YES*	**Social Bias**		*NO*
	Coordinates	Geometric	*YES*	**Proxemic**	Distance	*YES*
		Symbolic	*YES*		Dynamics	*NO*
	Relations	Metrical	*YES*	**Qualitative Relations**	Quantity Spaces	*NO*
		Ordinal	*YES*		Cardinal	*NO*
		Topological	*NO*		Grid-Based	*NO*
	Views	Allocentric	*YES*	**Naïve Operators**		*NO*
		Egocentric	*NO*			

Table 1.3.20: Environment and context in ICARE.

1.3.2.5 FAME - 2005

FAME, is a model-based Framework for Adaptive Multimodal Environments [Duarte et al. 2006] that provides a conceptual basis relating the different aspects of an adaptive multimodal system and a set of guidelines for conducting the development process. **FAME** provides an architecture designed to adapt multimodal applications to user actions, system events and environmental changes.

The **FAME** adaptive multimodal system architecture is composed of two parts: the internal and the external part. The internal part, is called Adaptation Module [**Figure 1.3.18**], which includes all the models needed for the adaptation mechanism and the Adaptation Engine (in orange) that is responsible for updating the models and generating the system actions.

This set of models (in violet) describe relevant attributes and behaviors regarding the user, the platform and the system environment.

To adapt the input and output modalities (in green), the information stored in these models is used; combined with the user inputs and the changes in the application state.

These models are also used to control the multimodal outputs and the presentation layout of the interface; the interaction possibilities available to the user; and how they are interpreted by the platform.

The external part corresponds to the multimodal layer. This includes the input/output devices and the *fusion* (in **blue** -dotted) and *fission* (in yellow -dotted) modules, the events generated by the application and the adaptive layout.

It is responsible for:

- the multimodal *fusion* of user inputs,
- the transmission of the events generated by the application to the adaptation core,
- the execution of the multimodal *fission* of the system actions, and
- determining the presentation's layout.

The multimodal *fusion* and *fission* components are shaped by the Adaptation Module: it influences both modules by determining the weight of each modality and the patterns of integration in the *fusion* or *fission* processes.

The multimodal *fusion* component is responsible for determining the intent of the user in every action from the information gathered by the different input modalities.

The choice of the weights can be determined either from the user collected data or environmental conditions. The user model could also be used to influence the operation of the multimodal *fusion* based on user preferences.

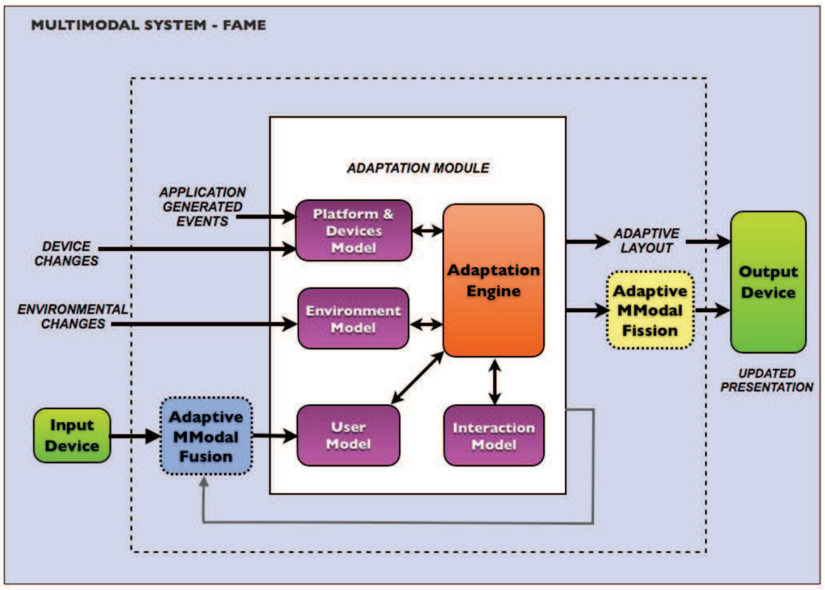


Figure 1.3.18: FAME Multimodal Architecture

The adaptation is based in three different classes of inputs:

- the user actions, issued from any of the input devices;
- the application-generated events and device changes, responsible for changing the state of any of the components participating on the interaction cycles; and finally,
- the environmental changes, acquired by sensors.

In short, the **FAME** Framework covers input and output processes without describing these processes as sensor or effector systems: it is limited to a device approach. *Fusion* and *Fission* are covered at the feature and the modality level [**Table 1.3.21**].

Direction	Both	*YES*	
Level of Abstraction	Input Abstraction	Device	*YES*
	Output Abstraction	Device	*YES*

Supporte d Modes	Acoustic	*YES*
	Visual	
	Haptic	
	Others	*NO*

Multiple Modalities for each Supported Mode ?	*YES*
Media Support	*YES*

Fusion	Data	*YES*
	Feature	
	Semantic	*NO*
	Hybrid	*NO*

Fission	Data	*YES*
	Modality	
	Semantic	

Table 1.3.21: Multimodal Characteristics in FAME.

The **FAME** Framework does not give any informations about a multimodal session or about the exchanges of events between the components. This architecture uses the information stored in several models for controlling the multimodal outputs of the interface and the interaction possibilities available to the user and how the are interpreted by the platform. The adaptation is guided by rules defined in a behavioral matrix[56] that express the output modality or combination of modalities in one of its dimensions, while in other dimension it can express how the information should be presented. Some of the rules in the matrix can be sequential (exclusive) and others can be simultaneous. These rules are used to improve the adaptation and to update the context by observing the history of the user behavior. Finally, multiple strategies to handle interaction are given: finite-state, frame-based, information-based and learning. [Table 1.3.22]

The adaptation rules are also part of the decisional support in the **FAME** Framework that helps to satisfy constrains, dynamically reduce options and recommend combinations based on the interaction history. The learning mechanism based on formal decision tables of rules extends the system, the domain and the real world knowledge.

Interaction (Dialog)	Multimodality	Sequential	*YES*	Interaction Types	Direct	*YES*
		Simultaneous	*YES*		Free flow	*NO*
		Composite	*NO*			
	Synchronized	*YES*		Focus Handled	*YES*	
	Historized	*YES*		State recorded	*YES*	
	Turn Taking	*YES*		Context updated	*YES*	
	Strategies	One	*NO*	Interpretation	by grammar	*YES*
					by structure	*YES*
		Multiple	*YES*		by planning	*YES*
					by learning	*YES*
					by intention int.	*NO*
Decision	Goal	Satisfy constraints	*YES*	Techniques	Formal	*YES*
		Reduce options			Non Formal	*NO*
		Recommend				
	Knowledge	System	*YES*	Uses	Resolve ambiguity	*YES*
					Uncertain reasoning	*NO*
		Domain	*YES*		Support interpretation	*YES*
					Collaborate in planning	*YES*
		General	*NO*		Support RW understanding	*YES*

Table 1.3.22: Management of Multimodal Behavior in FAME.

56 The detailed description of a behavioral matrix is out of the scope of this analysis. For more information see [Duarte et al. 2006]

While the output layout is decided, the adaptation mechanism takes into account factors such as the output capabilities of the device and the user preferences, characteristics and knowledge [**Table 1.3.23**].

With this goal, the Platform & Devices Model is used. It describes the characteristics of the platform of execution and of the devices attached to it. The Platform attributes relate to invariant characteristics of the platform while the Devices represent the software and hardware artifacts. Nevertheless the Device modeling describes only low level capabilities of devices, without any abstract descriptions or any multimodal informations.

Device Modeling	**At Model Level**		*YES*	**Abstract Description**		*NO*
	At Profile Level		*YES*	**Multimodal Info**		*NO*
	Managed	by Document	*YES*			
		by API	*NO*			
		by Service	*NO*			
		by Repository	*NO*			
User Modeling	**At Model Level**		*YES*	**Data collection**		*NO*
	At Profile Level		*YES*	**Personalization**		*YES*
	Generation	Deducted	*YES*	**Levels**	Sensorial	*YES*
		Inferred	*NO*		Behavioral	*YES*
					Semantic	*NO*
	Stored	Centralized	*YES*	**Stereotypes**		*NO*
		Distributed	*NO*	**Social support**		*NO*
Domain Modeling	**Application**		*YES*	**Interaction**		*YES*
	Presentation		*YES*	**Use**		*YES*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*YES*	**Abstract**		*YES*
				Taxonomies		*NO*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*NO*			

Table 1.3.23: Participants in FAME.

The User Model stores relevant user preferences and characteristics. This model may include physical attributes and preferences describing the preferred interface behavior and presentation characteristics. All this information is stored in a centralized model.

The Interaction Model describes information for domain modeling like the generic information about the availability of components for the presentation and the interaction. Each component has a set of templates, covering the broadest range possible of combination of devices, user groups and environments.

A set of templates of higher level organize the components available into a presentable version of the interface. These are component templates or composite templates. [**Table 1.3.24**]

The behavioral matrix is used to assist into the development of the adaptation rules. The matrix reflects the behavioral dimensions in which a user can interact with an adaptable component. It describes the adaptation rules, the pasts interactions and how the rules can be applied according to the usage behavior.

Each cell of the matrix holds a tuple with up to three elements. The first element corresponds to the current rules, the second counts the number of times the rule has been activated and the third defines a threshold for rule activation.

This matrix, with a set of guidelines, systematize the development process and models the domain, with each phase of analysis building upon previous phases. [**Table 1.3.24**] However, the emphasis is not placed on the temporal relations between modalities. Even if a module is identified in the architecture, this tool does not offer a specific solution for the combination of mechanisms of interaction. Thus, any combination can be defined as one of the behavioral dimensions of the matrix but no guide is proposed to identify different forms of possible combinations.

The Environment Model describes the characteristics of the situation of usage [**Table 1.3.24**] that can have an impact on the presentation and the interaction aspects of the application. For example, the ambient noise of the environment influencing both speech input and output modalities.

Usage Situation	At Model Level		*YES*	Relationships		*YES*
	At Profile Level		*YES*	Mental Models		*NO*
	Factual Description		*NO*	Social Qualities		*NO*
	Interpretation Description		*NO*			
Space-Time Situation	Ontologies		*NO*	Emotional Bias		*NO*
	Other Context Descriptions		*YES*	Social Bias		*NO*
	Coordinates	Geometric	*YES*	Proxemic	Distance	*YES*
		Symbolic	*YES*		Dynamics	*NO*
	Relations	Metrical	*YES*	Qualitative Relations	Quantity Spaces	*NO*
		Ordinal	*NO*		Cardinal	*NO*
		Topological	*NO*		Grid-Based	*NO*
	Views	Allocentric	*YES*	Naïve Operators		*NO*
		Egocentric	*NO*			

Table 1.3.24: Environment and context in FAME.

The Contribution of Fame for a Multimodal Architecture of Reference

To sum up, the **FAME** Framework addresses the context-awareness issues using models even though the content of these models is not described in detail. A behavioral matrix is proposed and some guidelines to describe rules are given but very concisely and devices or modalities are described without a generic approach. The main contribution for an architecture of reference is the effort to model environmental conditions with a learning mechanism and the use of context-awareness for the *fusion* and the *fission* mechanisms.

1.3.2.6 OPENINTERFACE - 2009

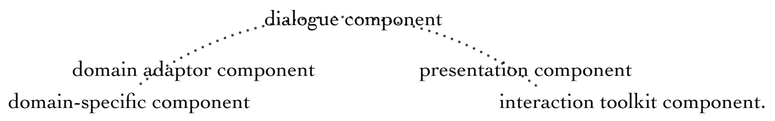
The **OPENINTERFACE** Platform [Serrano et al. 2009] is an open-source software solution designed to support fast prototyping and implementation of interactive multimodal systems. Its goal is to provide an evolvable software solution implementing a prototyping feature for multimodal interaction, independent in terms of devices, operative systems, interaction techniques and programming.

Architecture models allowed in OpenInterface

The platform adopts a modular architecture in which components are the principal type of objects. These modules do not constrain to the use of a component model, it allows MVC [Goldberg 1984], PAC [Coutaz et al. 1993] or ARCH [Arch 1992] implementations.

Nevertheless, the ARCH architecture is preferred in the studied papers through the use of ICARE as a starting point for this platform.[57]

57 The Arch model is also called the «Seeheim-revisited» model. In this model the interface is separated of the application (functional core) and the dialog controller in a component pattern forming an arch:



A Component is characterized by a) an API to communicate with its services; b) a packaging mechanism including installation/configuration procedures; c) its documentation an finally, d) its independence because a component must not make assumptions about the platform or features of other components outside the import procedure.

A Component provides a set of services/functionalities called Facets including input device drivers (in green), signal-processing (in green), *fusion* (in blue -dotted), *fission* (in yellow -dotted), networking, and graphical interfaces. These Facets have input/output interfaces called Pins. A pin can be a Sink, for receiving data; a Source for retrieving data; or a Callback for sending data and for importing external functionalities.

In order to build a running application [Figure 1.3.19] OpenInterface Pipelines are used. This is a mechanism of interconnection and configuration of components using the PDCL (Pipeline Description and Configuration Language) [Lawson 2006].

It allows the control over the components life-cycle and over the distribution at runtime, and provides data flow control at low level -e.g. threshold, filter- and high level -e.g. multicast, synchronization-. OpenInterface Pipelines also supports dynamic reconfiguration of connections at runtime.

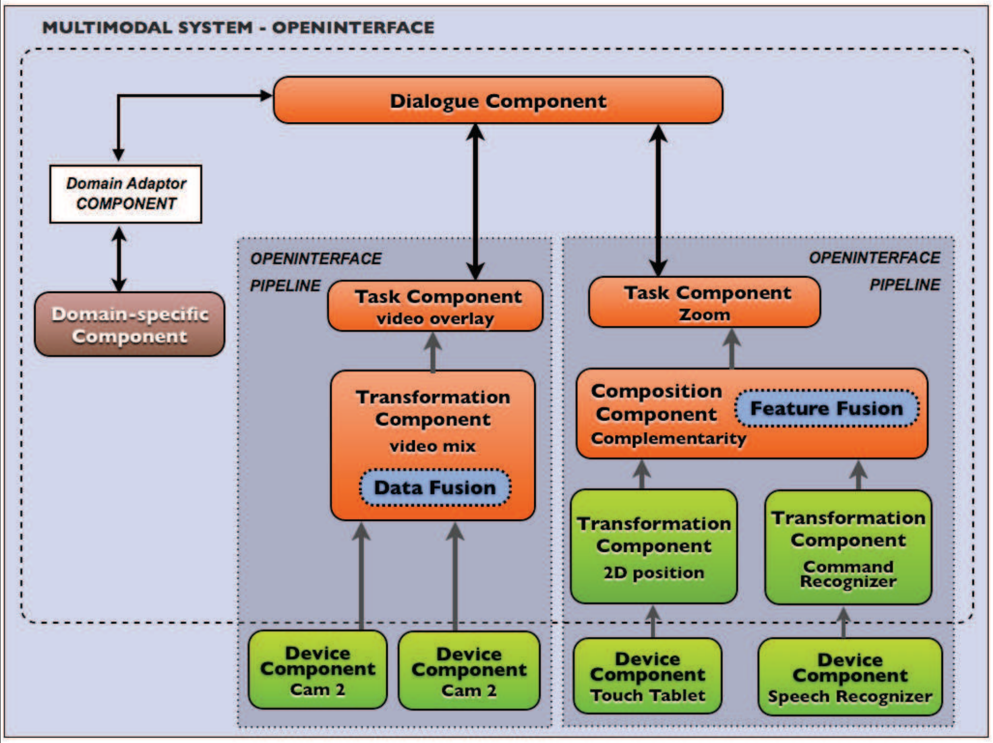

MULTIMODAL SYSTEM - OPENINTERFACE
Dialogue Component
Domain Adaptor COMPONENT
Domain-specific Component
OPENINTERFACE PIPELINE
Task Component video overlay
Transformation Component video mix
Data Fusion
Device Component Cam 2
Device Component Cam 2
OPENINTERFACE PIPELINE
Task Component Zoom
Composition Component Complementarity
Feature Fusion
Transformation Component 2D position
Transformation Component Command Recognizer
Device Component Touch Tablet
Device Component Speech Recognizer

Figure 1.3.19: OPEN INTERFACE Multimodal Architecture

The conceptual model of the **OPENINTERFACE** multimodal architecture presents a set of generic and tailored components. Generic components define reusable operations. Tailored components are components implemented in an ad-hoc way for a specific interaction technique or interactive application.This model also proposes three main types of components: devices, transformation chain and tasks.

As we can see in the **OPENINTERFACE** conceptual model, inputs are handled in a *Sensor System* abstraction while outputs are not covered. **[Table 1.3.25]**

This abstraction is defined as Device Components (in green) at the material side, and in the logical side, Pipelines, a workflow level prospect that shows the components and the conceptual links among them. As a result, both abstractions, Devices and Pipelines, compose a complex input Modality.

Direction	Input		*YES*
Level of Abstraction	Input Abstraction	System	*YES (Components & Pipelines)*
	Output Abstraction	System	*NO*

Supported Modes	Acoustic	*YES*	Multiple Modalities for each Supported Mode ?	*YES*
	Visual			
	Haptic			
	Others	*NO*	Media Support	*NO*

Fusion	Data	*YES*	Fission	Data	*NO*
	Feature			Modality	
	Semantic	*YES*		Semantic	
	Hybrid	*YES*			

Table 1.3.25: Multimodal Characteristics in OPENINTERFACE

The modality *fusion* is presented as a temporal synchronization based on Complementarity and Redundancy/Equivalence and is distributed within the platform in the Composition components.

As in ICARE platform, the *fusion* is proposed at all levels. **[Table 1.3.25]**

The multimodal session criteria is partially covered by the xml files written with the Pipeline Description and Configuration Language. In these files the assembly of components defined by describing the distribution, interconnection and configuration of the components. Nevertheless this is a static approach that does not need a dedicated manager. **[Table 1.3.26]**

Session	Supported	*YES*
	Handle by Manager	*NO*
	Migrated	*NO*
	Historized	*YES*

Interaction (Dialog)	Multimodality	Sequential	*YES*	Interaction Types	Direct	*YES*
		Simultaneous			Free flow	
		Composite				
	Synchronized		*YES*	Focus Handled		*YES*
	Historized		*NO*	State recorded		*YES*
	Turn Taking		*YES*	Context updated		*YES*
	Strategies	One	*NO*	Interpretation	by grammar	*YES*
					by structure	*YES*
					by planning	*YES*
		Multiple	*YES*		by learning	*NO*
					by intention int.	*NO*

Decision	Goal	Satisfy constraints	*YES*	Techniques	Formal	*YES*
		Reduce options			Non Formal	*YES*
		Recommend				
	Knowledge	System	*YES*	Uses	Resolve ambiguity	*YES*
					Uncertain reasoning	*YES*
		Domain			Support interpretation	*YES*
					Collaborate in planning	*YES*
		General			Support RW understanding	*YES*

Table 1.3.26: Management of Multimodal Behavior in OPEN INTERFACE.

The communication paradigms (event- based, remote procedure call, pipe, etc) are implemented with adapters/connectors, which are entities that mediate interactions among components by establishing rules. Consequently, the event handling is implementation-dependent and the only information given about events is its management with two types: instantaneous events and perduring events. **[Table 1.3.26]**

The interaction is handled using the CARE properties that covers the sequential, simultaneous and composite criteria. This approach covers also the free flow interaction with the four types of components: Task, Composition, Transformation and Device supporting multiple strategies of composition and temporal relations. [**Table 1.3.25**]

Concerning the Decision, the **OPENINTERFACE** Platform uses a set of formal rules for the control of the mechanism of modality combination. As a design tool the use of rules is not oriented to resolve ambiguity, uncertain reasoning or to support real world understanding. In contrast, as it is the case for ICARE, these rules support the input generation and interpretation and collaborate on planning in the authoring phase. [**Table 1.3.26**]

To take into account factors such as the input devices' capabilities [**Table 1.3.27**] the **OPENINTERFACE** Platform uses two languages to describe the Device components: the Component Interaction Description Language (CIDL) [Lawson 2006] and the Multimodal Component Description Language (MCDL) [Serrano 2010].

According to the CIDL a component must have: an id, a name, a programming language, a container (the description of the delivery format like a jar file, shared object library, archive or directory), a description of the I/O pins and their input and output functions, a description of the complex data types.

The MCDL allows to describe the component according to the type of the **OPENINTERFACE** component, its behavioral dependance and the technical details of the physical device. This refers to the device modeling approach, which is made also at a profile level, with the multimodal abstraction covered by the MCDL document and stored in repositories. [**Table 1.3.27**]

Device Modeling	**At Model Level**	*YES*		**Abstract Description**		*YES*
	At Profile Level	*YES*		**Multimodal Info**		*YES*
	Managed	by Document	*YES*			
		by API	*NO*			
		by Service	*NO*			
		by Repository	*YES*			
Domain Modeling	**Application**	*YES*		**Interaction**	*YES*	
	Presentation	*NO*		**Use**	*YES*	
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*YES*	**Abstract**	*YES*	
				Taxonomies	*YES*	
		Design Spaces	*YES*	**Performance Semantics**	*NO*	
		Temp. Relations	*YES*			

Table 1.3.27: Participants in OPENINTERFACE.

On the other hand, the domain modeling is very advanced, coming from the conceptual work around the interaction classification and analysis. Application, interaction, use, and activity granularity are covered by the Pipelines mechanism and the classification of components. As an authoring tool, the application life-cycle is completely covered. There is a lack of high level semantics mostly due to its focus on the input side of interaction with a sensor system perspective. Turn taking, interaction feed-back between modalities or interaction management are not detailed.

Finally, the Composition components covers the temporal situation with its implementation based on temporal relations and the CARE composition types [**Table 1.3.28**].

Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*YES*	**Temporal Operators**		*YES*

Table 1.3.28: Environment and context in OPENINTERFACE

The Contribution of Fame for a Multimodal Architecture of Reference

To sum up, the main contributions of the **OPENINTERFACE** Platform for a reference architecture in multimodal systems comes from the domain modeling. The management of well defined task in the classification of components, the distribution of fusion processes and the definition of tasks as a control layer abstraction playing an intermediary role is very useful for our purpose of generalization of roles.

1.3.2.7 SQUIDY - 2009

SQUIDY [König et al. 2010] is a common interaction library and a software architecture that enables the unification of a great variety of very heterogeneous device drivers and special-purpose toolkits.

This tool consists of three logical parts: a) the Squidy Manager that provides a flexible infrastructure for data processing, transport, management, and persistence; b) the Squidy Designer, a GUI for interactive visual programming of Squidy Pipelines and visual feedback; and c) the Squidy Client Implementations that are external applications executed on a client platform and providing input data to the Squidy Manager.

The Pipes & Filters architectural pattern

The conceptual architecture of the **SQUIDY** framework is a multi-threaded Pipes and Filters architecture [Allen et al. 1992] where each component has a set of inputs and a set of outputs. A component reads streams of data on its inputs and produces streams of data on its outputs, delivering a complete instance of the result in a predefined order.

This is usually accomplished by applying a local transformation to the input streams and computing incrementally so output begins before input is consumed.

In the Pipes and Filters model components are called 'filters' while in **SQUIDY** they are named Nodes and the connectors 'Pipes' serve as conduits for the streams, transmitting outputs of one filter to inputs of another [**Figure 1.3.20**].

Pipes, Nodes, Pipelines and the Workspace are derived from the generic entity Processable which incorporates methods to do any kind of processing using threads to permit asynchronous and independent execution of each Pipeline or Pipeline segment.

The Pipeline acts as a container for the processing chain which normally consists of Nodes and Pipes. Multiple instances of Pipelines can be created and connected by Pipes and it can contain other Pipelines.

This principle allows to group functionality in nested structures. A Pipeline behaves like a Node if it is observed from the group perspective.

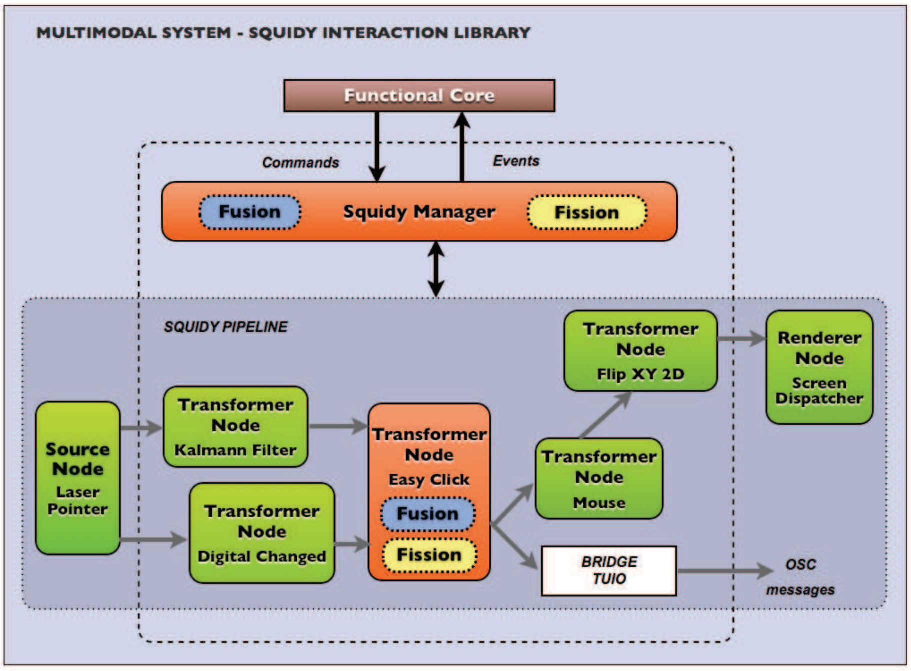

MULTIMODAL SYSTEM - SQUIDY INTERACTION LIBRARY
Functional Core
Commands
Events
Fusion
Squidy Manager
Fission
SQUIDY PIPELINE
Source Node Laser Pointer
Transformer Node Kalmann Filter
Transformer Node Digital Changed
Transformer Node Easy Click
Fusion
Fission
Transformer Node Mouse
Transformer Node Flip XY 2D
Renderer Node Screen Dispatcher
BRIDGE TUIO
OSC messages

Figure 1.3.20: SQUIDY Multimodal Architecture

SQUIDY manages Nodes for data processing which are connected via Pipes to form Pipelines. The internal properties of a Node are not integrated directly into its data processing Pipeline. Every Node provides just a single pair of input and output ports including automatic data type management. The data flow within the Pipeline chain is unidirectional. Several instances of Squidy Manager (in orange) can connect to each other using a Node called Squidy Remote. This Node represents a Bridge which translates between Squidy data objects and the Open Sound Control protocol [Freed et al. 2009]. The OSC protocol consists of plain ASCII text and can be transmitted as a data stream via network interfaces. This concept enables distributed processing of input data.

The Squidy Manager provides the necessary infrastructure for Nodes and data processing in the Pipeline. Like the Filter Graph Manager in DirectShow[58], it controls all Nodes contained in the Node graph (Pipeline). Tasks are accomplished by this management instances. Each instance of Squidy Manager uses the system time of the platform on which it is executed to create time stamps.

Direction	Both		*YES*
Level of Abstraction	Input Abstraction	System	*YES*
	Output Abstraction	System	*YES*

Supported Modes	Acoustic	*YES*	Multiple Modalities for each Supported Mode ?	*YES*
	Visual			
	Haptic			
	Others	*NO*	Media Support	*NO*

Fusion	Data	*YES*	Fission	Data	*YES*
	Feature			Modality	
	Semantic	*NO*		Semantic	*NO*
	Hybrid	*NO*			

Table 1.3.29: Multimodal Characteristics in SQUIDY.

58 Windows API available at http://msdn.microsoft.com/en-us/library/ms783323.aspx

In a distributed environment, if accurate synchronization is required, time-stamping of all participants can be accomplished by an external application or service.

Multimodal Characteristics [**Table 1.3.36**] in **SQUIDY** are handled in a high level of abstraction. The framework supports input and output at a *sensor system* and *effector system* level. Multiple modalities are supported thanks to the included library of Nodes and Bridges. Media is not supported and the application turn taking mechanisms or the pipeline coordination by the Squidy Manager are not explained.

Even if the Pipelines can be viewed like multimodal session entities, they are not defined as a relation between a user and a group of resources allocated to the interaction, but like a processing chain and they are mostly unimodal. The event handling is not detailed.

Sequential and simultaneous interaction are supported in **SQUIDY** pipelines by ensuring synchronized and direct interaction. Turn taking between pipelines is application-dependent and handled by the Squidy Manager but this mechanism is not described. Focus is handled by the processing chain with the data flow management. The state of the Pipeline and the nodes is recorded by the Squidy Manager and represented by the user interface of the designer tool in design-time but no information is given about the management of the state on runtime. [**Table 1.3.30**] On the other hand, Decision support is not addressed by **SQUIDY**.

Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous	*YES*		Free flow	*NO*
		Composite	*NO*		Both	*NO*
	Synchronized	*YES*		**Focus Handled**	*YES*	
	Historized	*NO*		**State recorded**	*YES*	
	Turn Taking	*NO*		**Context updated**	*NO*	
	Strategies	One	*YES*	**Interpretation**	by grammar	*YES*
					by structure	*YES*
		Multiple	*NO*		by planning	*YES*
					by learning	*NO*
					by intention int.	*NO*

Table 1.3.30: Management of Multimodal Behavior in SQUIDY.

Device modeling is handled by descriptions given at low level and in a component type high level: user defines the logic of an interaction technique by choosing the desired nodes from a collection -knowledge base- and connecting them in an appropriate order assisted by a node suggestion based on heuristics[**Table 1.3.31**].

Data type hierarchy in **SQUIDY** is based on the primitive virtual devices taxonomy proposed by [Wallace 1976]. Any data consists of single or combined instances of these generic data types [**Figure 1.3.21**]. Each of them consists of a type-specific aggregation of atomic data types such as numbers, strings or Boolean values bundled by their semantic dependency.

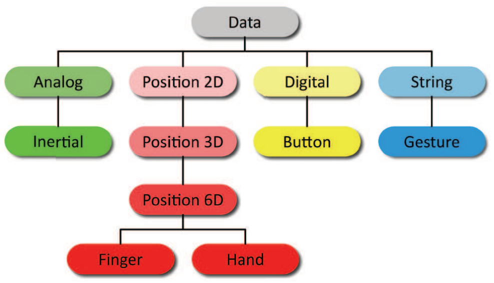


Figure 1.3.21: SQUIDY generic dataTypes

Domain modeling also helps for this task. The application and the interaction are modeled with the Pipeline mechanism. It provides a technical advantage, since the Node functionality is contained in "black-boxes" that allow information encapsulation, modifiability and high reuse.

Device Modeling	**At Model Level**		*YES*	**Abstract Description**		*YES*
	At Profile Level		*YES*	**Multimodal Info**		*NO*
	Managed	by Document	*YES*			
		by API	*NO*			
		by Service	*NO*			
		by Repository	*YES*			
Domain Modeling	**Application**		*YES (pipelines)*	**Interaction**		*YES (pipelines)*
	Presentation		*NO*	**Use**		*NO*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*YES*	**Abstract**		*YES*
				Taxonomies		*NO*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*NO*			

Table 1.3.31: Participants in SQUIDY.

Finally the possibility for multiple input and output connections offers a high degree of flexibility and the potential for massive parallel execution of concurrent nodes.

No User modeling, Usage situation modeling or Spatio-temporal situation for the multimodal application are proposed. In contrast, the temporal situation [**Table 1.3.32**] is covered by the process of synchronization underneath the Pipeline abstraction and the data processing approaches.

Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*NO*	**Temporal Operators**		*NO*

Table 1.3.32: Environment and context in SQUIDY.

To sum up, even though the **SQUIDY** Library does not address the context-awareness issues, and neither proposes models. Devices and modalities are described with a generic approach coming from the Nodes abstraction and the generic data types presented. The main contribution for a Multimodal Architecture of Reference is the effort to model the task flow in Pipelines and the distribution of the *fusion* and the *fission* mechanisms over transformer Nodes. The multiple instances of the Squidy Manager and the Pipeline imbrication are also some interesting contributions of this proposal.

The Contribution of Squidy for a Multimodal Architecture of Reference

1.3.3 Architectures from Applications

Outline Our goal for the following section is to confront our grid of optionals modules and features in multimodal systems with the empirical models coming from the study of a set of implementations: multimodal applications, multimedia players or authoring tools. The main difference with the proposals already visited is that this implementations focuses more in concrete aspects of the management of modalities (like their distribution, intelligent behavior or composition) than in the definition and validation of a generic model of multimodality.

By analyzing each approach under the light of these generic criteria, we will bring out the possible contributions of each approach against the goal of a context-aware Multimodal Architecture of Reference, for large-scale networks like the Internet. We suppose that by analyzing their distribution of features and responsibilities we can find some common information about some generic components for such an architecture.

1.3.3.1 MEDITOR - 1995

MEDITOR [Bellik 1995] is a text editor for blind people. The system handles four input devices: a speech recognition system, a braille keyboard, a normal keyboard and a mouse. In the output side it handles a braille display, a speech synthesis module and a screen.

MEDITOR addresses the information *fusion* problem, and the importance of the temporal factor for it with an architecture model built around independent interpreters and a central dialogue controller which uses decision rules to ensure *fusion*. The multimodal application was created following the SPECIMEN architecture model, which is designed to integrate an interaction model, oriented to support language processing, with an interaction model oriented to support physical actions. For this reason SPECIMEN is based on augmented transition networks (dialog oriented) and at the same time, also based on message handling (event oriented).

In this architecture every input or output is represented by un object (in green) that ensures the homogeneous communication between the generators of messages and the input/output devices. [**Figure** 1.3.22] For example, the Input Messages Generator (in orange) can demand to the input object if a recognition task is in progress, and the Output Messages Generator (in orange) can demand to the output object if the current task of vocal synthesis is finished.

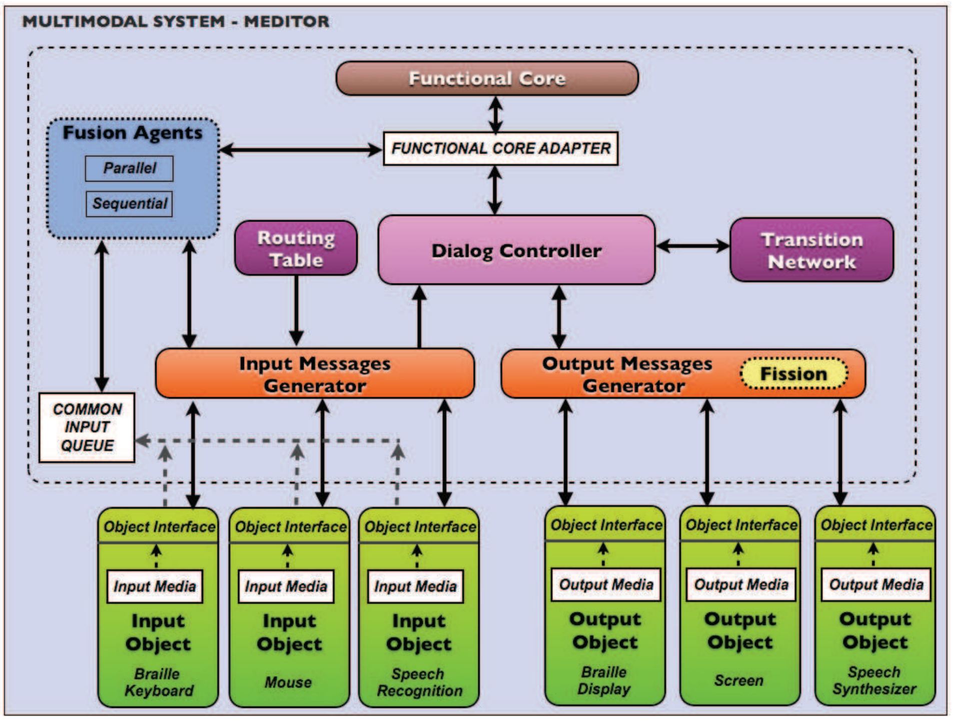


Figure 1.3.22: MEDITOR's Multimodal Architecture

A common chronological queue of input messages manages the history and the state of the exchanges in the system while SPECIMEN also allows the specification of messages using operators for composition.

These composition operators facilitate the description of sequential and / or parallel actions with two kinds of *fusion* agents (in **blue** -dotted): the sequential *fusion* agents (AFS), responsible to produce messages when a predefined temporal series of interactions are detected (one agent by predefined series) and the parallel *fusion* agents (AFP), responsible to produce messages when a set of predefined interval sequences[59] of time is detected (one agent by predefined overlapping sequence). [**Figure 1.3.22**]

A routing table (TA in **violet**) [**Figure 1.3.22**] keeps track of pointers to the agents responsible of the detection of each message. The input messages generator (GME) takes a message from the input queue, sends the message to the associated agent based on the routing table and then, it produces input messages to transmit to the dialog controller.

The dialog controller (CD in **pink**) [**Figure 1.3.22**] manages the state of the augmented transition networks. It decides which is the more adapted process to execute according to the received messages and the conditions predefined in the options to evaluate. For example it activates the actions described in the transition arcs.

The output messages generator (GMS in **orange**) [**Figure 1.3.22**] is responsible to compose the output messages assuring the scheduling of sequential messages and the synchronization of simultaneous messages. These output messages can force the dialog controller to wait the end of the output process (synchronous, blocking message) or can allow the dialog controller to continue its processing (asynchronous, not blocking message).

With the inputs and outputs handled as Media Objects **MEDITOR** embraces an approach with an abstraction that views both at a system level [**Table 1.3.33**].

Direction	**Both**	*YES*	
Level of Abstraction	**Input Abstraction**	System	*MEDIA OBJECT*
	Output Abstraction	System	*MEDIA OBJECT*

Supported Modes	Acoustic	*YES*	**Multiple Modalities for each Supported Mode ?**	*YES*
	Visual	*YES*		
	Haptic	*YES*		
	Others	*NO*	**Media Support**	*YES*

Fusion	Data	*YES*	**Fission**	Data	*YES*
	Feature	*YES*		Modality	*YES*
	Semantic	*NO*		Semantic	*NO*
	Hybrid	*NO*			

Table 1.3.33: Multimodal Characteristics in MEDITOR.

The *fusion* mechanism is proposed at the event level covering *data fusion* and *feature fusion*. Semantic and hybrid *fusion* are not treated. The output messages generator is responsible of the *fission* and the distribution of sequential and parallel messages to the output media at the data and modality level. [**Table 1.3.33**]

In **MEDITOR** the multimodal session is not addressed. Events are supported and handled by three specialized managers: the input messages queue, the input messages generator and the output messages generator. [**Table 1.3.34**]

The temporal order is ensured, a monomodal and multimodal support is provided by the fusion agents and disambiguation is handled with a routing table and predefined rules. Event granularity is restricted to the message level and the event management does not update the context information. [**Table 1.3.34**]

59 A overlapping multi-set of time points in more than one semantic dimension. See supra part 1.2.3.2 Temporal Situation (Temporal series and sequences).

The *fusion* agents handle sequential and simultaneous multimodality for direct and free flow interaction. The interaction is synchronized with the intervention of the *fusion* agents and the Dialog Controller; and historized with a routing table. Turn taking is handled by the activity of the transition network and the Dialog Controller. Multiple strategies are used in *fusion* agents for interpretation. [**Table 1.3.34**]

Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*YES*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*YES*
	Disambiguated		*YES*	Associated to context		*NO*
	Multiple Granularity		*NO*			
Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous			Free flow	
		Composite	*NO*			
	Synchronized		*YES*	**Focus Handled**		*YES*
	Historized		*YES*	**State recorded**		*YES*
	Turn Taking		*YES*	**Context updated**		*NO*
	Strategies	One	*NO*	**Interpretation**	by grammar	*YES*
		Multiple	*YES*		by structure	*YES*
					by planning	*YES*
					by learning	*NO*
					by intention int.	*NO*
Decision	**Goal**	Satisfy constraints	*YES*	**Techniques**	Formal	*YES*
		Reduce options			Non Formal	*YES*
		Recommend	*NO*			
	Knowledge	System	*YES*	**Uses**	Resolve ambiguity	*YES*
		Domain			Uncertain reasoning	*YES*
		General			Support interpretation	*YES*
					Collaborate in planning	*YES*
					Support RW understanding	*NO*

Table 1.3.34: Management of Multimodal Behavior in MEDITOR.

Decision is handled by the Dialog Controller to satisfy constraints and reduce options, this is made with formal and non formal techniques for all uses excepting the better understanding of the real world surrounding the user which is not covered. [**Table 1.3.34**]

In **MEDITOR** devices are modeled as media objects, following a basic taxonomy. This model is managed as a class-oriented API with multimodal information in properties used in the composition of messages. However, this description is only based on concrete properties (low level). [**Table 1.3.35**]

MEDITOR implementation is intended to show with an specific case the importance of the multimodality handled at low level (message fusion) for the intelligent reconstruction of an user interface. The problem of an intelligent adaptation of the interface to a blind user is out of the scope of this implementation. For this reason, User Modeling is not addressed.

For the same reason, **MEDITOR** does not address the issue of a domain model for the application needs and semantics, or an use model only concerning the emotionally-centered interaction. An interaction model is provided with the description of the dialog controller activity. This activity uses an Interaction model to make decisions based on the transition networks and the results of the *fusion* agents in the composition of messages.

Design and runtime phases are covered by the description of the **MEDITOR** implementation. The Interaction granularity selected for this approach is the action and the sequence, leaving aside the interaction based on the mean (semantic level). One of the most important efforts of the SPECIMEN architecture and the **MEDITOR** implementation is to describe and manage temporal relations between messages with some two time-based categories of interaction: sequential and parallel. [**Table 1.3.35**]

Device Modeling	**At Model Level**		*YES*	**Abstract Description**		*NO*
	At Profile Level		*YES*	**Multimodal Info**		*YES*
	Managed	by Document	*NO*			
		by API	*YES*			
		by Service	*NO*			
		by Repository	*NO*			
Domain Modeling	**Application**		*NO*	**Interaction**		*YES*
	Presentation		*YES*	**Use**		*NO*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*NO*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*YES*	**Abstract**		*NO*
				Taxonomies		*YES*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*YES*			

Table 1.3.35: Participants in MEDITOR.

MEDITOR does not address the semantic facets of multimodality. As a result the Usage situation is not covered. In contrast, the temporal situation is mostly the principal issue treated by the implementation, which is reflected by the proposals for almost all the temporal criteria, excepting the management of semi-intervals of time. In contrast, no information is given about the spatial situation affecting the interaction. [**Table 1.3.36**]

Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*YES*	**Temporal Operators**		*YES*

Table 1.3.36: Environment and context in MEDITOR.

To sum up, **MEDITOR** can cooperate in various aspects for the definition of a Multimodal Architecture of Reference. First, by the proposal of a component responsible of the state of the system as a registry or state manager: the routing table. Secondly, the definition of an entity responsible to make decisions based on data stored in a model: the dialog controller. The work around event handling is also a very complete proposal that shows the importance of the transport layer in a *multimodal system*. And finally the challenge to incorporate two different models of interaction in a more holistic multimodal system, capable of supporting discrete interaction processing and continuos interaction processing.

The Contribution of Meditor for a Multimodal Architecture of Reference

1.3.3.2 REA -1999

Embodied Conversational Agents are a kind of *multimodal system*, in which several modalities must be integrated into one representation of the speaker intention, or into one representation of the predefined intentions of the humanoid.

Description of Embodied Conversational Agents

Yet, a specificity of conversational agents is that in these systems the interpretation is carried out while the user is executing an utterance (*semantic fusion*) and not only until the user has finished the utterance (*feature fusion*) .

With this goal, these agents focus in notions like the «discourse structure» or «the conversation» that takes into account the reason for using a verbal or a non verbal device in a particular situation or a phase of the conversation.

REA [Cassell et al. 1999] is one of these computer-generated humanoids that represents an intelligent system in its user interface, and that allows the real-time interaction between a conversational agent and a human user. The metaphor of this interaction is a conversation between **REA** and a potential real-state customer.

REA is a humanoid with verbal and nonverbal behaviors (gaze, facial expressions, posture and hand gestures). **REA** can also detect the user signs in nonverbal turn-taking speech (gestures made by the user while talking) and thus interrupt to let the user intervene. It has a fully articulated graphical body, can sense the user passively through cameras and audio input, and is capable of speech with intonation, facial display, and gestural output. To enhance the realism of the interaction, the agent and its 3D environment are displayed on a big screen, putting **REA** in a human scale.

The architecture of **REA** allows to detect in the user discrete behaviors oriented to interrupt the conversational flow and to demand the speaking turn. It also detects gestures used to emphasize the conversational content. As a result, this information is merged with the verbal content.

This architecture proposes the separation of content and process: a special generation module is dedicated to verbal and nonverbal behavior generation, taking an abstract representation of the communicative intent and giving it a form according to the rules of social interaction (face-to-face). This modularization and the use of a formal model of representation, allows to explore the automatic generation of multimodal behavior.

REA's architecture follows the FMTB model, in which the goals concerning the interaction (action goals) and the goals concerning the conversational content (proposition goals) are conveyed by functions that participate in the generation of multimodal outputs (humanoid behaviors) and the interpretation of multimodal inputs (user behaviors).

In **REA** modalities are assimilated to devices and integrated by a single conceptual representation: the Device (input/output) component (in green). This abstract representation, has slots for interactional (control and command) and propositional information (content-oriented). [**Figure** 1.3.23]

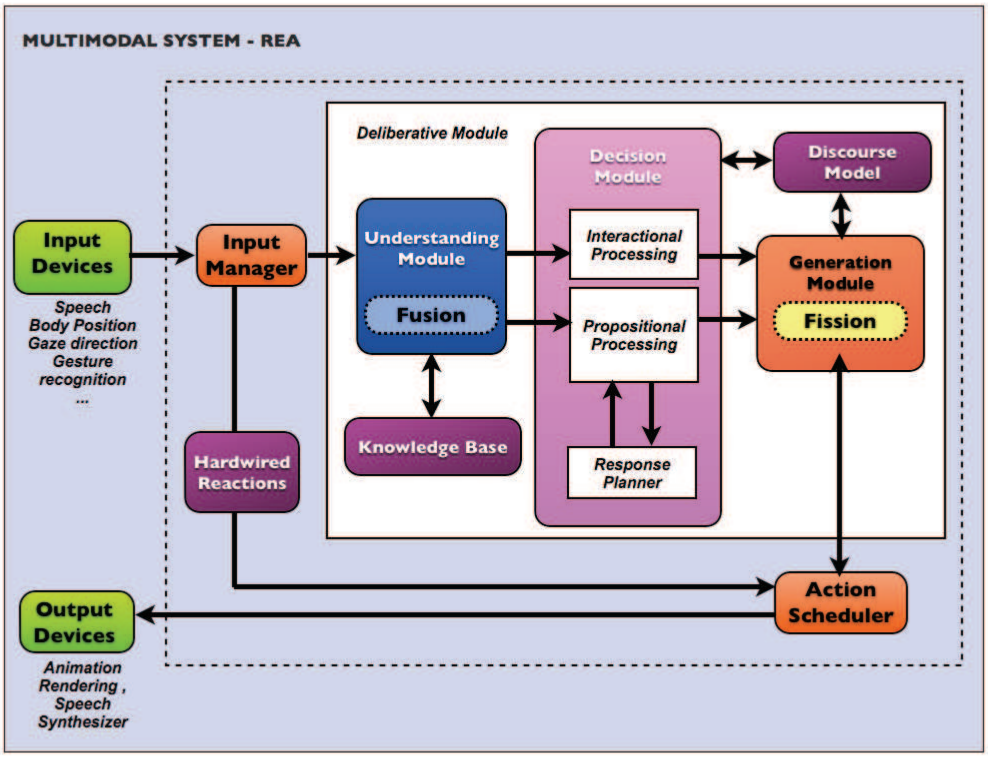


Figure 1.3.23: REA Multimodal Architecture

On the input side, the data sent by the input Device to the Input Manager (in **orange**) is time stamped with start and end times in milliseconds. The synchronization is made using NTP (Network Time Protocol) clients. This synchronization is key for associating verbal and nonverbal behaviors. [**Figure 1.3.23**]

The *multimodal system* used for **REA** allows the embodied conversational agent to watch for feedback and turn taking requests, while the human send these at any time through various modalities.

REA tracks these different threads of communication in an appropriate way for each thread with different response-time requirements: some, such as feedback and interruption, occur on a sub-second time scale while other processes are focused on activities at a different timescale.

On the output side, the multimodal and real-time requirements demands a perfect coordination between speech and nonverbal behavior such as gesturing. It is resolved with the implementation of the motor skill mechanism.

As we can see, the decision is centralized in a deliberative module containing the Understanding Module (in **blue navy**), the Decision Module (in **pink**) and the Generation Module (in **orange**) while the management of the behavior with the Input Manager and the ActionScheduler (in **orange**) is put in the periphery. [**Figure 1.3.23**]

This processing module for deliberative discourse handles all input that requires a discourse model for proper interpretation, which includes many of the interactional behaviors as well as all propositional behaviors.

The Understanding Module (in **blue navy**) is responsible of the *fusion* process. It integrates all input modalities into a coherent understanding of what the user is doing and for translating a set of behaviors into a function, interactional or propositional.

The Decision Module (in **pink**) processes the interactional acts (to control and command the interaction) and the propositional acts (those that contribute to the content). [**Figure 1.3.23**]

The Interactional Processing submodule updates the interaction cycle (conversation) state, whether an interaction cycle has started, who has the turn, and whether the interaction has been put on hold while the user momentarily do something else.

The Propositional Processing submodule chooses the adequate responses to the propositional input (e.g. find answers to questions) and communicates with the Response Planner if necessary.

The Response Planner submodule is responsible for formulating sequences of actions, even when this actions need to be executed during future execution cycles.

The Generation Module (in **orange**) is responsible for composing a set of coordinated actions and sending the actions to the Action Scheduler (in **orange**) for the output performance.

This architecture implies a sequential processing [**Figure 1.3.23**]. The Input Manager collects input from all modalities and decides whether the data requires an instant reaction (e.g. a discrete output event) or deliberative discourse processing in (e.g. a continuos output like a streamed phrase composed of a lips animation and a sound). Hardwired Reaction handles quick reactions to stimuli. These stimuli then provoke a modification of the agent's behavior (like a facial change output) without much delay.

In this way, the Input Manager is responsible to obtain data from the various input Devices, convert it into a form usable by other modules in the system and route the results to the Understanding Module.

Interactional information (control and command) can be also forwarded directly to the Reaction Module to minimize the response time of the *multimodal system*. [**Figure 1.3.23**] This component is hard-wired directly to the output process, avoiding the decisional part of the system.

These hardwired reactions enable the character to respond immediately to certain unimodal user inputs that require fast reaction but do not require any inferencing or reference to the discourse model. In REA a hardwired reaction is, e.g. tracking the user's location with the character's eyes.

Finally, the Action Scheduler is responsible of the *fission* mechanism, by scheduling motor events to be sent to the animated figure representing the agent. It also synchronizes actions at the lowest level across modalities, for example, the gesture stroke and the pitch peak in speech and prevent collisions between competing motor events. The Action Scheduler takes a set of modality-specific commands and executes them in a synchronized way through the use of event conditions specified for each output action which define when the action should be executed.

All these modules communicate using the Knowledge Query and Manipulation Language (KQML) [Chalupsky et al. 1992], a communication protocol intended to support interoperability among intelligent agents in a distributed application. The principal premise of KQML is that to make agents understand each other they have to not only speak the same language, but also have a common ontology. An ontology is a part of the agent's knowledge base that describes what kind of things an agent can deal with and how they are related to each other. Every message carries the following metadata: type (speech act type), qualifiers (keywords list), content-language (a language name), content-ontology (the ontologies assumed in the content), content-topic (topic of the knowledge given in the ontology) and content (the content of the actual sentence).

The multimodal characteristics of **REA** are depicted in [**Table 1.3.37**]. The system covers both input and output directions. The abstraction used is at the level of Device, which means that inputs/outputs are not addressed as independent systems with a high level of abstraction. Three modes are covered with multiple modalities but without a generic reflection about the media engaged in the interaction. The *fusion* and the *fission* are addressed at all levels, and because of its conversational nature, with special emphasis on semantics.

Direction	**Both**	*YES*	
Level of Abstraction	**Input Abstraction**	Device	*INPUT DEVICE*
	Output Abstraction	Device	*OUTPUT DEVICE*

Supported Modes	Acoustic	*YES*
	Visual	*YES*
	Haptic	*YES*
	Others	*NO*
Multiple Modalities for each Supported Mode ?		*YES*
Media Support		*NO*

Fusion	Data	*YES*
	Feature	*YES*
	Semantic	*YES*
	Hybrid	*YES*
Fission	Data	*YES*
	Modality	*YES*
	Semantic	*YES*

Table 1.3.37: Multimodal Characteristics in REA.

On the other hand, in **REA** all the facets of the management of the multimodal behavior are pointed out [**Table 1.3.38**]. The notion of «conversation» as an interaction cycle can be perceived by analogy as our multimodal session criteria. Some of the key features of a conversation in **REA** that are similar to the characteristics of our multimodal session criteria and event handling are:

- the distinction between propositional (content) and interactional (command and control) functions of conversation
- the use of several modalities
- the need of a precise identification and cooperation of participants.
- the importance of timing among behaviors (and the increasing co-temporality or synchrony among participants)
- the distinction between conversational behaviors (the interaction performance) and conversational functions (the interaction modality or combination of modalities).

Multimodal events are handled in **REA** at all levels, with special attention on the association with the context of the conversation and the multiple granularity of events.

Session	Supported		*YES*			
	Handle by Manager		*YES*			
	Migrated		*NO*			
	Historized		*YES*			
Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*YES*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*YES*
	Disambiguated		*YES*	Associated to context		*YES*
	Multiple Granularity		*YES*			
Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous	*YES*		Free flow	
		Composite	*YES*			
	Synchronized		*YES*	**Focus Handled**		*YES*
	Historized		*YES*	**State recorded**		*YES*
	Turn Taking		*YES*	**Context updated**		*YES*
	Strategies	One	*NO*	**Interpretation**	by grammar	*YES*
					by structure	*YES*
		Multiple	*YES*		by planning	*YES*
					by learning	*NO*
					by intention int.	*YES*
Decision	**Goal**	Satisfy constraints	*YES*	**Techniques**	Formal	*YES*
		Reduce options	*YES*		Non Formal	*YES*
		Recommend	*YES*			
	Knowledge	System	*YES*	**Uses**	Resolve ambiguity	*YES*
					Uncertain reasoning	*YES*
		Domain	*YES*		Support interpretation	*YES*
					Collaborate in planning	*YES*
		General	*YES*		Support RW understanding	*NO*

Table 1.3.38: Management of Multimodal Behavior in the REA.

The interaction and dialog are studied as sequential and simultaneous phenomena. [Table 1.3.38] They are handled by multiple formal and non formal strategies with a focus on the interpretation of the user intention and the realistic presentation of the humanoid intent, in order to perform the conversation in a natural way.

As an intelligent *multimodal system*, **REA** covers all the criteria for Decision handling, excepting the support of real world understanding, given that human behavior analysis or reality mining are not its goal. [Table 1.3.38]

Concerning the management of participants in **REA** [Table 1.3.39] the devices are treated only at the profile level with a common message protocol and an API but without any specific model or dedicated properties to represent the multimodal information.

The user is modeled with a focus on personalization, with a discourse and conversational models (in **violet**) that are not inferred from a data collection but rather from a predefined user model which is deducted in the data processing.

This model covers a) sensorial data like the support for the interpretation of voice tonality, b) behavior data like the synthesis of eye raising and c) semantic data like the negotiation of speaking turns. This model is stored in a centralized discourse model.

The multimodal domain is largely addressed by **REAs** architecture that integrates the real-time multimodal aspects with the deep semantic generation and the multimodal synthesis features that we can found in animated conversation.

Action, sequence and mean are covered in all the life-cycle phases of the multimodal application. Nevertheless, the interaction phenomenon is studied only from the perspective speech acts and categories. No taxonomy or abstract classification is given, and interaction design spaces for this embodied conversational agent are not suggested. The temporal relations between modalities is not described.

In contrast, **REA** covers in a very complete way the semantics of the interaction performance, on the input side with the meaning interpretation of the body gestures of the user and in the output side, with the composition of modalities guided by the expression of emotions of the humanoid. [**Table** 1.3.39]

Device Modeling	**At Model Level**		*NO*	**Abstract Description**		*NO*
	At Profile Level		*YES*	**Multimodal Info**		*NO*
	Managed	by Document	*NO*			
		by API	*YES*			
		by Service	*NO*			
		by Repository	*NO*			
User Modeling	**At Model Level**		*YES*	**Data collection**		*NO*
	At Profile Level		*YES*	**Personalization**		*YES*
	Generation	Deducted	*YES*	**Levels**	Sensorial	*YES*
		Inferred	*NO*		Behavioral	*YES*
					Semantic	*YES*
	Stored	Centralized	*YES*	**Stereotypes**		*NO*
		Distributed	*NO*	**Social support**		*NO*
Domain Modeling	**Application**		*YES*	**Interaction**		*YES*
	Presentation		*YES*	**Use**		*YES*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*YES*
	Interaction Classification	Categories	*YES*	**Abstract**		*NO*
				Taxonomies		*NO*
		Design Spaces	*NO*	**Performance Semantics**		*YES*
		Temp. Relations	*NO*			

Table 1.3.39: Participants in REA.

Environment and context are treated in **REA** at three levels. The Usage situation is reflected by the conversational behaviors and the discourse model abstractions. [**Table** 1.3.40]

The interpretation of gestures, mental models and user intent are included on the context information. The social qualities of the interaction domain are treated by the adopted protocol and its use of mental models based on the sharing of ontologies and other knowledge supports.

The temporal Situation is reflected by the synchronization mechanisms in the Input Manager and the Action Scheduler implemented in **REA.** [**Table** 1.3.40]

Time is expressed as points and intervals in a valid, transactional and symbolic way. The propositional acts of the conversation handles homogenous temporal data and the interactional acts handles mostly heterogeneous temporal data, even if sometimes it can also use homogeneous temporal data.

This temporal data is viewed under the light of their temporal relations, for example, with the turn taking management or the hardwired reactions. Nevertheless no description about the use of temporal operators is given.

Finally, as a static implementation (**REA** does not implies nomadic interaction) the spatio-temporal aspects are less present than in other types of multimodal systems.

In **REA** these criteria are treated mostly for the proxemics issues raised by the context of the conversation and its consequences for the responses and intentions of the humanoid. Also for this reason, the system has an affective and emotional bias, based on descriptions in the knowledge and discourse models. [**Table 1.3.40**]

Usage Situation	**At Model Level**		*YES*	**Relationships**		*YES*
	At Profile Level		*YES*	**Mental Models**		*YES*
	Factual Description		*YES*	**Social Qualities**		*YES*
	Interpretation Description		*YES*			
Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*YES*	**Temporal Operators**		*NO*
Space-Time Situation	**Ontologies**		*NO*	**Emotional Bias**		*YES*
	Other Context Descriptions		*YES*	**Social Bias**		*NO*
	Coordinates	Geometric	*YES*	**Proxemic**	Distance	*NO*
		Symbolic	*NO*		Dynamics	*NO*
	Relations	Metrical	*YES*	**Qualitative Relations**	Quantity Spaces	*NO*
		Ordinal	*YES*		Cardinal	*NO*
		Topological	*YES*		Grid-Based	*NO*
	Views	Allocentric	*YES*	**Naïve Operators**		*NO*
		Egocentric	*YES*			

Table 1.3.40: Environment and context in REA.

The Contribution of REA for a Multimodal Architecture of Reference

To sum up, the main contributions of **REA** for a reference architecture in multimodal systems comes from its emotional intelligence and its architectural model. Some of these contribution are :

- the management of well defined tasks in the definition of components with a large emphasis in decision making,
- the separation between decision and behavior
- the use of social and cultural knowledge by the means of shared ontologies,
- the proposal of hardwired behaviors to reduce response time,
- the support all over the system of the separation between control data and content data,
- the effort to handle in the interaction cycle -and with the same mechanism, the exchange of content (by the management of modalities in the interaction function) and the semantics of the actions/interaction (by the management of conversational behaviors)
- and finally, the use of an emotional bias to enrich the user model and to control the *fusion* and the *fission* processes of the multimodal interaction.

1.3.3.3 MACROMEDIA DIRECTOR - 1997

MACROMEDIA DIRECTOR [Adobe 2011] was a cross-platform authoring environment for multimodal and multimedia applications; widely used on cd-roms, dvd's, e-learning, and cultural projects.

The scope of this authoring tool is different to the already studied frameworks; this is a proprietary software for which the goal is not necessarily to produce implementations resulting from a multimodal research. However, at the time, it was a de facto tool largely used; and proved to be a successful tool to produce multimedia and multimodal applications. For this reason, its architecture deserves to be studied in our research.

The **MACROMEDIA DIRECTOR**'s architecture is based on the MVC model, where a component, called the Score (in orange) represents the controller, the Cast (in violet) represents the data model and the Stage (in turquoise) represents the presentation state. This structure allows the designers to configure multimodal applications following this pattern with the use of a metaphor coming from the film production industry.

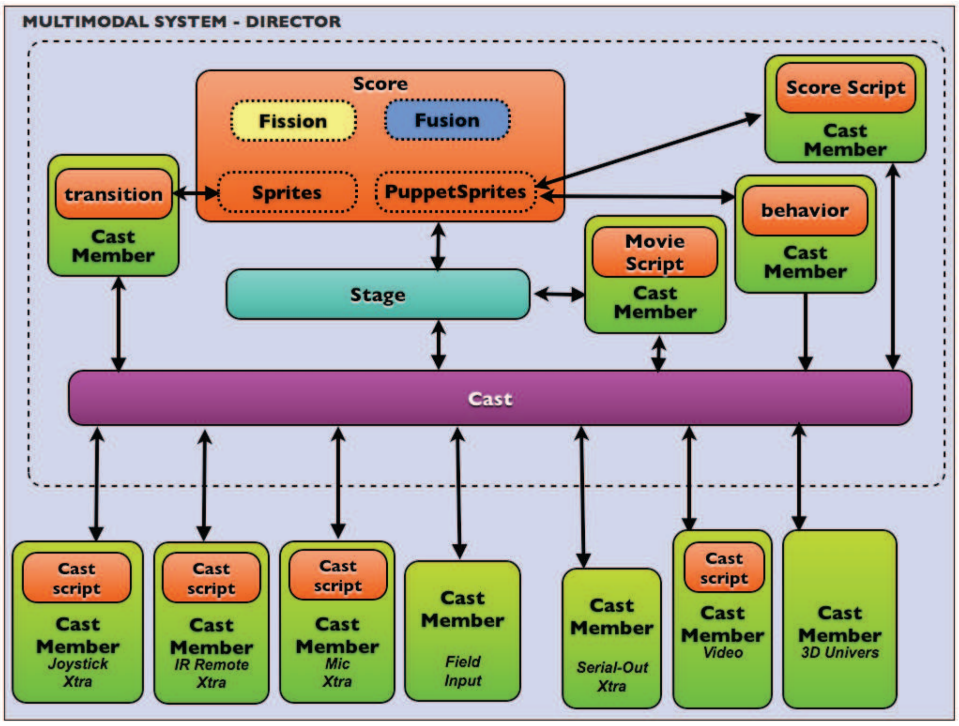


Figure 1.3.24: MACROMEDIA DIRECTOR Architecture

The Cast (in violet) is a registry for all of the elements in the application (called the Movie). This elements are represented by a specific kind of component, the Cast Member (in green), that can be an input/output object, but also an element to control the interaction, like the scripts, behaviors and transitions. [**Figure 1.3.24**] A Cast Member can be also an Xtra (a kind of plug-in) which is a form of standalone code: it contains extra procedures and functions that can be stitched on the fly to extend the movie capabilities, for example to handle IR inputs, serial microControllers or user-defined inputs of any kind. Typical uses are to provide support for specialized hardware and to give access to wayward bits of the operating system. Finally Cast Members can carry Cast Scripts (in orange) that do not appear as separate cast members themselves, but only as adjuncts to the input/output they are attached to. In general they respond only to events passed to the Cast Member they are attached to.

The Stage is the component responsible to present the composed view and to keep track of its state. It keeps an internal list of Sprites, which are the channels allowed to the content. It is the support for the animation, listing the Cast Member instances over a spatio-temporal representation of the interaction cycle. [**Figure 1.3.24**]

The Score controls the interaction and the presentation. It handles the instructions describing how the various Cast Members crop up in the movie and change and interact over time. It is a Pipeline connecting the Cast Member instances and supporting the control of all the multimodal interaction. It supports the *fusion* and *fission* mechanisms in the Movie by its management of the presentation space and time, the scripting features and the binding mechanism of sprites instances with the Cast. The Score also allows the modalities layout by *fission,* composing a presentation over time. The default execution is just to do all the things shown in the Score straight through, one after another according to the disposition of contents in each Sprite (channel) and in some cases, with a transition[60] between the content in the Sprites. [**Figure 1.3.24**]

However, if the Sprite (channel) is declared as a «puppet» Sprite, it can be controlled by a Cast Member dedicated to this task. If the puppet property of a sprite is true, it will behave as instructed by scripts or behaviors. If false, the sprite will behave according to the default directions in the Score. These control cast members (in **orange**) are of three types: a behavior, a Score Script or a Movie Script.

- A behavior is an object containing code that can be associated to any Sprite for handling the events related to the Sprites' specific class.

- A Score script is an object containing code for handling the events for a defined state of the Sprite at a precise time of the execution Pipeline. Handlers contained within them are only accessible to events restricted in time and space to the frames in which they appear in the Pipeline.

- A Movie Script is a code that handles the global state of any Sprite and any moment of the execution. Movie scripts and the handlers they contain are globally accessible - they can be called from anywhere in the movie

The principal difference between these script types is the scope, what events typically are sent to them, and the order in which events are received. Events normally pass through scripts in the order specified in the «Lingo Messaging Hierarchy» stopping as soon as they find a script with a handler that can use them. This handler can then, «pass» the event to the next handler, in a bubbling-like mechanism.

The basic multimodal characteristics in **MACROMEDIA DIRECTOR** are ensured by various means: the Cast Members objects are abstractions that can be extended by behaviors and cast scripts, to support specific events and communicate with the application's environment by using the API's of the host Operating System. By the use of Xtras, multiple *modes* and *modalities* can be supported. Media control, *fusion*, *fission* and categorizations are given as API's. However, semantic or hybrid levels are not covered. [**Table 1.3.41**]

Direction	**Both**	*YES*

Level of Abstraction	**Input Abstraction**	System	*CAST MEMBER*
	Output Abstraction	System	*CAST MEMBER*

Supported Modes	Acoustic	*YES*	**Multiple Modalities for each Supported Mode ?**	*YES*
	Visual	*YES*		
	Haptic	*YES*		
	Others	*NO*	**Media Support**	*YES*

Fusion	Data	*YES*	**Fission**	Data	*YES*
	Feature	*YES*		Modality	*YES*
	Semantic	*NO*		Semantic	*NO*
	Hybrid	*NO*			

Table 1.3.41: Multimodal Characteristics in DIRECTOR.

Concerning the multimodal session, the information about the state of the movie is handled by the Stage component but it can not be migrated between movies neither historized. [**Table 1.3.42**]

60 A transition is a Cast Member containing a filter to apply to all the Sprites in the Movie.

The Events are supported and handled by the Score at the level of pipeline execution (continuos) and at the level of scripting (discrete). With the same mechanisms the interaction is synchronized and the turn taking supported. The state of the interaction is recorded with the Score component. Two interaction strategies are supported: finite-state (grammars) and frame-based structures (task graphs and type hierarchy). [**Table 1.3.42**]

Session	Supported		*YES*			
	Handle by Manager		*YES*			
	Migrated		*NO*			
	Historized		*NO*			
Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*YES*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*YES*
	Disambiguated		*NO*	Associated to context		*NO*
	Multiple Granularity		*YES*			
Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous	*YES*		Free flow	*NO*
		Composite	*NO*			
	Synchronized		*YES*	**Focus Handled**		*NO*
	Historized		*NO*	**State recorded**		*YES*
	Turn Taking		*YES*	**Context updated**		*NO*
	Strategies	One	*NO*	**Interpretation**	by grammar	*YES*
					by structure	*YES*
		Multiple	*YES*		by planning	*NO*
					by learning	*NO*
					by intention int.	*NO*

Table 1.3.42: Management of Multimodal Behavior in DIRECTOR.

The management of the multimodal system's Participants in **MACROMEDIA DIRECTOR** covers Device modeling mostly through the models and profiles accessible in the multiple API's provided by the tool, and organized in media categories and types. [**Table 1.3.43**] User modeling is not addressed, while Domain modeling is based on the application (movie) and presentation approaches in all the phases of the multimodal system life-cycle. The granularity of the interaction is covered only as actions (events) and sequences (sprites in the score). A high level classification of the interaction is not modeled: only a taxonomy related to the Cast Member media APIs is given. [**Table 1.3.43**]

Device Modeling	**At Model Level**		*YES*	**Abstract Description**		*NO*
	At Profile Level		*YES*	**Multimodal Info**		*NO*
	Managed	by Document	*NO*			
		by API	*YES*			
		by Service	*NO*			
		by Repository	*NO*			
Domain Modeling	**Application**		*YES*	**Interaction**		*NO*
	Presentation		*YES*	**Use**		*NO*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*NO*	**Abstract**		*NO*
				Taxonomies		*YES*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*NO*			

Table 1.3.43: Participants in DIRECTOR.

Environment and context are treated in **MACROMEDIA DIRECTOR** from the Temporal Situation and the Space-Time Situation, while the Usage Situation is not treated.

The Temporal Situation is covered in almost all facets with the Score and its temporal APIs excepting the semi-interval abstraction, the temporal relations or the temporal operators.

On the other hand, with the Stage component and some of the APIs included in the behavior components, the space-time situation is covered at the sensorial level, which means that generic coordinates with metrical and ordinal relations can be described and used.

Quantity spaces and the cardinal relations also are proposed to control the Sprites in the presentation at the runtime phase. The space-time is only covered from an allocentric perspective. [**Table 1.3.44**]

Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*NO*	**Temporal Operators**		*NO*
Space-Time Situation	**Ontologies**		*NO*	**Emotional Bias**		*NO*
	Other Context Descriptions		*NO*	**Social Bias**		*NO*
	Coordinates	Geometric	*YES*	**Proxemic**	Distance	*NO*
		Symbolic	*NO*		Dynamics	*NO*
	Relations	Metrical	*YES*	**Qualitative Relations**	Quantity Spaces	*YES*
		Ordinal	*YES*		Cardinal	*YES*
		Topological	*NO*		Grid-Based	*NO*
	Views	Allocentric	*YES*	**Naïve Operators**		*NO*
		Egocentric	*NO*			

Table 1.3.44: Environment and context in DIRECTOR.

The Contribution of Director for a Multimodal Architecture of Reference

To sum up, **MACROMEDIA DIRECTOR** can collaborate in the definition of possible features for a Multimodal Architecture of Reference with its structure mixing the management of continuos interaction with a Pipeline mode, and the Event-driven control of discrete interaction (and free flow interaction) defined with the use of puppetSprites.

Another contribution is the mechanism of control and support of *fusion* and *fission* in the Score component, implemented with APIs in the Cast Members.

Finally, we can point out also as a contribution, the implementation and description of a component responsible of the global state of the multimodal/multimedia system (the Score) and its interface with a global registry (the Cast).

1.3.3.4 SMARTKOM - 2003

SMARTKOM is a multimodal dialog system that combines speech, gesture, and mimics at the input and the output levels. [Herzog et al. 2004] One of its major scientific goals is to design new computational methods for the seamless integration and mutual disambiguation of multimodal input and output on a semantic and pragmatic level.

In **SMARTKOM** the user delegates a task to a virtual communication assistant, visualized as a non-realistic character on a graphical display. Both communication partners collaborate during a problem solving process in which the assistant agent accesses the background services and presents results on the output channels.

SMARTKOM implements a *situated* interpretation of possibly ambiguous or incomplete multimodal inputs and the composition of coordinated, cohesive, and coherent multimodal outputs. With this goal it is based on a multi-blackboard architecture with parallel processing threads that support the media fusion and media design processes.

The **SMARTKOM** components are [**Figure** 1.3.25]:

- Interface modules: audio module, gesture device (input), display manager, audio (output).
- Recognizers and synthesizers: gesture recognizer (input), prosody and speech recognizers (input), speech synthesizer and the character animator (output).
- Semantic processing modules: gesture and speech analysis, media fusion, intention recognition, discourse and domain modeling, action planning, presentation planning, and concept-to-speech generation. (to create meaning representations or transform them)
- External services: the function modeling module (interface to external services like EPG databases, map services and web information crawlers)
- A Dynamic Help module (analysis of the situation and the context).
- The System Watchdog (monitors the system's state).
- The Lexicon (a dynamic knowledge source)

Each component corresponds to an independent process since **SMARTKOM** has been implemented as a distributed system.

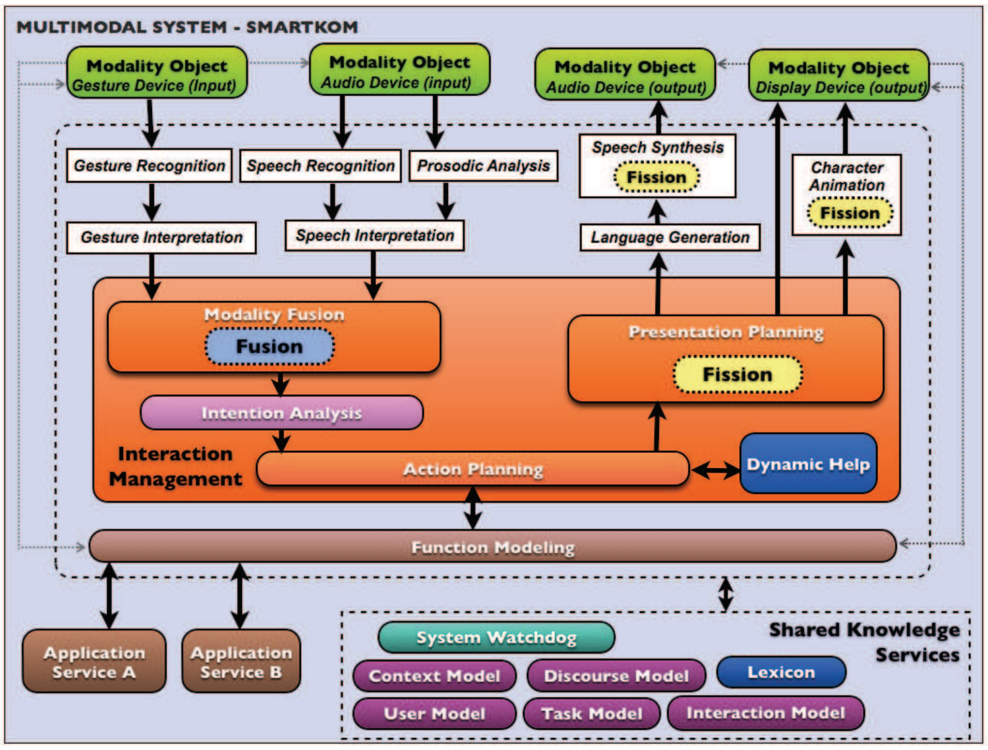


Figure 1.3.25: SMARTKOM Multimodal Architecture

The architectural model of **SMARTKOM** is the publish- subscribe style, focused on the communication processes: the interaction is event-driven using communication links between a message sender, who acts as an event source (data producer), and a set of recipients (data consumers). Thus, the components communicate via data pools that correspond to named message queues.

All the multimodal inputs and output in **SMARTKOM** use a common representation approach, which allows the definition of a generic interaction model. The basic structure is a three-tiered representation model where for each object in any input and output modality a Modality Object is stored for that particular event. Such an object refers to a Discourse Object located at the discourse level. Input and output from multiple modalities can therefore contribute to the same Object at the discourse level.

The input/output processing (that is sensor-specific) is separated into distinct components, which connect a technical device to the *multimodal system* and which encapsulate access to the underlying hardware. By the use of Modality Objects (in **green**) it provides a generic and hardware-independent interface (at the signal level or at the symbol level) for further processing.

A set of modality-specific components are responsible of the analysis of input data. It consist of a group of recognizers (represented with a white color code in [**Figure 1.3.25**]) that transform sensor signals from the environment into symbolic information and analyzers that provide a meaning description for these signals.

The Interaction Module (in **orange**) constructs and maintains the model of human-computer interaction. It uses input sources to calculate values describing very specific aspects of the data. These values (indicators) are combined to construct more general values (models). The general values are then passed on to other modules that can exploit them to support the interaction.

The first task of the Interaction Management is the Modality Fusion (in **blue** -dotted). This stage merges the meanings coming from different modalities into one unified meaning representation that reflects a mutual disambiguation of multimodal input at the semantic and pragmatic levels. [**Figure 1.3.25**]

In the Intention Analysis (in **pink**), the intention of the user is identified, and the next steps to be taken are decided. The intention recognizer ranks the interpretation hypotheses coming from the Modality Fusion and selects the most likely one based on the discourse model, which is then passed on to the Action Planner (in **orange**). In this way, the final ranking becomes highly context-sensitive. [**Figure 1.3.25**]

The Action Planner(in **orange**) has the task to coordinate the actions to perform. It identify the action that the user expects from the *multimodal system* on the basis of the interpretation of the incoming intentions. Then Action Planner interacts with the various applications (in **brown**): a) it selects the appropriate application for the user's request, b) it requests the chosen application and finally c) it forwards the response data (e.g. the presentation content) to the Presentation Planning component (in **orange**) which is responsible to transform the response data into coordinated multimodal output with a modality *fission* mechanism. It includes sub-tasks like content selection, media and modality allocation, layout design, and coordination. The planner applies predefined presentation strategies that decompose the complex presentation goal into presentation tasks. For example, the media fission strategy decides whether a description is to be uttered verbally or graphically. It is based on constraints in the strategies and the data in the context and the result is a hierarchical plan.

The behavior of the planner results in a uniform approach in planning, performing intention-based interaction. The input and output structures in all channels (and in all components) are based on a uniform representation: the use of dialogue acts like request, response and inform to describe the interaction exchanges. At the heart of the planing modules is a backward chaining, non-linear regression planning approach.

Separate modules for Function Modeling (in **brown**) responsible of information coercion mediate also the communication with the applications. It hides the details specific to the applications from the action planners. [**Figure 1.3.25**]

The Dynamic Help component (in **blue navy**) performs an analysis of the situation and the context with the planning component supervising this task. This top-level component guides the behavior of the Dynamic Help component in an abstract way. It determines how lower levels must compute specifications from knowledge sources and from the situation. [**Figure 1.3.25**]

Finally, **SMARTKOM** implements a monitoring mechanism for the *multimodal system* which is used to provide the presentation module with information about the current state of the internal processing.

It requires awareness of the system's capabilities, processing state and dialog situation and is used to provide immediate feedback or to initiate helpful reactions in case of processing problems. If the processing gets stuck the System Watchdog (in **turquoise**) component is able to detect this condition and to react with an appropriate feedback. It handles problematic events that have a negative property, e.g. internal errors or malfunctions, problems in the evolution of the dialog, changes of emotional states of the user, underspecified requests that cannot be resolved by discourse analysis, or indirect and vague help requests and information queries on the meta level. This contextual awareness is related to a set of models used by **SMARTKOM:**

- A multimodal Discourse Model (in **violet**): describing the semantic and pragmatic interpretation during input and output processing. It is dynamically updated as system output progresses and performs contextual reasoning and scoring.
- A context Model (in **violet**): handling references to situation parameters like current place and time.
- An Interaction Model (in **violet**): describing available modalities and user preferences for specific forms of communication as well as the affective state of the user to dynamically adapt the behavior of the system
- A Lexicon (in **blue navy**): a dynamic knowledge source, which is updated with additional lexical entries depending on dynamic application data as it is received from the external information services to process natural language input and output.

The communication between the distributed components within the **SMARTKOM** are based on the exchange of structured data through XML [W3C-XML 1996] messages coded with the MultiModal Markup Language (M3L) [Herzog et al. 2004]. It represents the information that flows between the components and in particular, M3L represents the information segmentation, synchronization and confidence in the processing results.

Multimodal ontologies in Smartkom

The main schema, describing the intentions of both the user and the system, is defined off-line in an ontology with more than 700 concepts and about 200 relations, which describe the abstract objects needed to communicate. For example, the planner's interface is almost exclusively based on the modeling of the intentions as defined in the ontology. In this way, conceptual taxonomies provide the foundation for the representation of domain knowledge as it is required within the *multimodal system* to enable a natural interaction. This ontology is created as a support for a closed world reasoning – everything the user and the system can talk about is encoded in the ontology. However, the basic information flow from inputs to outputs continuously adds further processing results so that the representational structure is refined step-by-step.

Direction	**Both**	*YES*	
Level of Abstraction	**Input Abstraction**	Device	*MODALITY OBJECT*
	Output Abstraction	Device	*MODALITY OBJECT*

Supported Modes	**Acoustic**	*YES*
	Visual	*YES*
	Haptic	*YES*
	Other	*NO*

Multiple Modalities for each Supported Mode ?	*YES*
Media Support	*YES*

Fusion	**Data**	*YES*
	Feature	*YES*
	Semantic	*YES*
	Hybrid	*YES*

Fission	**Data**	*YES*
	Modality	*YES*
	Semantic	*YES*

Table 1.3.45: Multimodal Characteristics in SMARTKOM.

The multimodal characteristics in **SMARTKOM** are implemented with the perspective of a Device with the Modality Object and for the input and the output direction. Three modes are supported with multiples modalities and *fusion* and *fission* are covered in all levels with a special focus on semantics. [**Table 1.3.45**]

The notion of a multimodal session is not described in **SMARTKOM.** In contrast, the multimodal event is supported in all the facets covered by our criteria list. [**Table 1.3.46**] It is described in the communication via data pools of named message queues and specified with the MultiModal Markup Language (M3L). Events are associated to context with the participation of the System Watchdog.

Interaction is managed by the Interaction Management Module and its subcomponents like the Modality Fusion, the Action Planning and Presentation Planning, with a special focus on the use of multiple strategies where the most relevant are planning, learning and intention interpretation.

Decision is used to achieve all the goals described by our criteria with the use of multiple techniques to solve all the types of problems raised by the multimodality management. [**Table 1.3.46**]

Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*YES*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*YES*
	Disambiguated		*YES*	Associated to context		*YES*
	Multiple Granularity		*YES*			
Interaction (Dialog)	**Multimodality**	Sequential	*YES*	**Interaction Types**	Direct	*YES*
		Simultaneous	*YES*		Free flow	
		Composite	*YES*			
	Synchronized		*YES*	**Focus Handled**		*YES*
	Historized		*YES*	**State recorded**		*YES*
	Turn Taking		*YES*	**Context updated**		*YES*
	Strategies	One	*NO*	**Interpretation**	by grammar	*YES*
					by structure	*YES*
		Multiple	*YES*		by planning	*YES*
					by learning	*YES*
					by intention int.	*YES*
Decision	**Goal**	Satisfy constraints	*YES*	**Techniques**	Formal	*YES*
		Reduce options	*YES*		Non Formal	*YES*
		Recommend	*YES*			
	Knowledge	System	*YES*	**Uses**	Resolve ambiguity	*YES*
					Uncertain reasoning	*YES*
		Domain	*YES*		Support interpretation	*YES*
					Collaborate in planning	*YES*
		General	*YES*		Support RW understanding	*YES*

Table 1.3.46: Management of Multimodal Behavior in SMARTKOM.

For the management of the participants in the multimodal system **SMARTKOM** proposes multiple models. Devices are modeled with the Modality Objects feature and this model is completed with the information stored on the Interaction Model and the Task Model.

The Device Model is an abstract description used by the APIs, completed by metadata in taxonomies and application services. [**Table 1.3.47**]

The User is modeled from static data but also updated with inferred data in the Interaction Model. The model describes sensorial, behavioral and semantic information about the user to personalize the interaction cycle.

These models are distributed. Stereotypes or the Social information are not addressed. [**Table 1.3.47**]

The Interaction Domain is described by a dedicated model: the multimodal Discourse Model that covers the interaction at the action, sequence and mean levels, the presentation and the use (from the perspective of the intention and emotion). The lifecycle of the application is not described. The description is abstract, based on taxonomies with a classification based on categories. To animate the humanoid or interpret the user intention performance semantics are used. [**Table 1.3.47**]

Device Modeling	**At Model Level**		*YES*	**Abstract Description**		*YES*
	At Profile Level		*YES*	**Multimodal Info**		*YES*
	Managed	by Document	*YES*			
		by API	*YES*			
		by Service	*YES*			
		by Repository	*NO*			
User Modeling	**At Model Level**		*YES*	**Data collection**		*YES*
	At Profile Level		*YES*	**Personalization**		*YES*
	Generation	Deducted	*YES*	**Levels**	Sensorial	*YES*
		Inferred	*YES*		Behavioral	*YES*
					Semantic	*YES*
	Stored	Centralized	*NO*	**Stereotypes**		*NO*
		Distributed	*YES*	**Social support**		*NO*
Domain Modeling	**Application**		*YES*	**Interaction**		*YES*
	Presentation		*YES*	**Use**		*YES*
	Life-cycle	Design	*NO*	**Granularity**	Action	*YES*
		Load	*NO*		Sequence	*YES*
		Run	*YES*		Mean	*YES*
	Interaction Classification	Categories	*YES*	**Abstract**		*YES*
				Taxonomies		*YES*
		Design Spaces	*NO*	**Performance Semantics**		*YES*
		Temp. Relations	*NO*			

Table 1.3.47: Participants in SMARTKOM.

The environment and context in **SMARTKOM** is handled with the Usage situation modeled describing relationships and mental models with a description of the interpretation. The Temporal Situation is handled as points and intervals in the Context Model and in all components with the . It represents the valid time, the transaction time and the symbolic time. Data is distributed homogeneously and heterogeneously. Temporal relations or temporal operators are not described. [**Table 1.3.48**]

The space-time situation is described with the Context Model with ontologies and categories. Coordinates are metrical and ordinal with an allocentric view. The Emotional bias is covered by the interpretation of the user's intention and the reproduction of the humanoid intentions. [**Table 1.3.48**]

The Contribution of Smartkom for a Multimodal Architecture of Reference

To sum up, **SMARTKOM** can contribute to a Multimodal Architecture of Reference in multiple facets. First, the distributed nature of the system, event-based and oriented to parallel processing can be an interesting approach when a multimodal system needs to be modular and extensible.

Second, **SMARTKOM** follows the principle that there should not be processing and presentation without an explicit representation. Based on this premise, it uses declarative data models coupled with a mechanism to infer these models from indicators. In **SMARTKOM** not only complete representations are passed between the functional blocks in the system; and for this reason, the support of incremental processing is very important.

Another contribution is the representation of the interaction cycle with a high level analysis, based on intentions described semantically and unified in a specific model (in this case the discourse model). Components like the Intention Analyzer, the Dynamic Help component and the System Watchdog participates in an interesting manner to maintain the coherence of the interaction according to this model.

Finally, the last contribution to highlight is the proposal of a semantic-based system, guided by well defined ontologies at all the levels: device, data and user modeling, communication, data processing, data restitution, assistance, failure/error handling and context awareness.

Usage Situation	**At Model Level**		*YES*	**Relationships**		*YES*
	At Profile Level		*YES*	**Mental Models**		*YES*
	Factual Description		*NO*	**Social Qualities**		*NO*
	Interpretation Description		*YES*			
Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*YES*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*NO*	**Temporal Operators**		*NO*
Space-Time Situation	**Ontologies**		*YES*	**Emotional Bias**		*YES*
	Other Context Descriptions		*YES*	**Social Bias**		*NO*
	Coordinates	Geometric	*YES*	**Proxemic**	Distance	*YES*
		Symbolic	*YES*		Dynamics	*NO*
	Relations	Metrical	*YES*	**Qualitative Relations**	Quantity Spaces	*NO*
		Ordinal	*YES*		Cardinal	*NO*
		Topological	*NO*		Grid-Based	*NO*
	Views	Allocentric	*YES*	**Naïve Operators**		*NO*
		Egocentric	*NO*			

Table 1.3.48: Environment and context in SMARTKOM.

1.3.3.5 GPAC - 2003

GPAC (Gpac Project on Advanced Content) [Le Feuvre et al. 2007], is a multimedia framework compliant to the MPEG-4 Systems standard [ISO-MPEG4 1998] used for research in multimedia, with a focus on graphics, animations and interactivity technologies.

The framework is composed of three main parts: a multimedia packager, several servers and a multimedia player, that we will analyze from the perspective of its support for multimodal applications.

The **GPAC** multimedia player can reproduce any video or audio format and can support multiple delivery protocols: it renders audiovisual content mixed with 2D and/ or 3D contents.

Until now, in our study of multimodal systems, we treated the architecture of a system as a collection of components together with a description of the interactions between these components -its connectors.

Graphically speaking, this lead us to a view of an abstract description of architectures as a diagram in which the nodes represent the components and the arcs represent the connectors. Color codes are used to make more easy to recognize the components that can have similar responsibilities or behaviors.

In this way, as we has already show in the precedent figures, connectors can represent interactions as varied as procedure calls, event broadcast, database queries, or pipes.

This illustration pattern reflects how up until now, we have presented some architectures keeping on mind the following questions:

What is the structural pattern—the components, connectors, and constraints? What is the underlying computational model?

What are the essential invariants of the structure?

What are some of the common specializations?

Following this method, **GPAC** can be described as an heterogeneous architecture.

First, as an audiovisual media player (supporting playback) compliant to the MPEG-4 Systems standard, it is a layered system, organized hierarchically, with each layer providing services to the layer above it and serving as a client to the layer below. This characteristic is a result of its relation with the MPEG4/DMIF[61] paradigm.

In the MPEG4/DMIF systems, inner layers are accessible only to the adjacent outer layer, except for certain functions carefully selected for export. As a result, in this system, the components generally implement some virtual machine in a high layer in the hierarchy. On the other hand, the connectors are defined by the protocols that determine how the layers will interact.

The **GPAC** player architecture follows this kind of layered design and relies on extensions which makes it highly flexible.

The player can participate in a client-server structure, in which all multimedia description tools are independent from the transport and the coding of the objects, whether vectorial (text, graphics) or rasterized (images, audio, video).

Media decoding is ensured for the most common media formats through modules containing the code provided by third-parties open-source projects.

Nevertheless, as a rich media player (supporting rich interaction), it can be seen also as a Pipes and Filters architecture, where each component has a set of inputs and a set of outputs, it reads streams of data on its inputs and produces streams of data on its outputs, delivering a complete instance of the result in a standard order. [**Figure 1.3.26**]

In this architectural pattern, the components are called "filters" while the connectors, called "pipes", serve as conduits for the streams, transmitting outputs of one filter to inputs of another.

In this section we will focus on this second aspect of the player for the client side, leaving aside some aspects concerning the multimedia systems architecture (e.g. the audiovisual media compression and broadcast), and the server side implementation, which are out of the scope of our study.

The entry point of a **GPAC** Player instance is the Terminal component. In MPEG-4, the terminal is linked to one or several MPEG-4 servers and the application content and the application description has to be streamed to the Terminal client from a predefined URL.

Thus, a Terminal (in **green**) is the abstract representation of a device . [**Figure 1.3.26**]

Every Terminal instance, contains several functional blocks: a Compositor (in **orange**), an Object Manager (in **violet**), a networking services layer (in **blue**), a Download Manager (Downloader-in **blue**), a Media Manager and a Root Scene (in **turquoise**). [**Figure 1.3.26**]

61 The Delivery Multimedia Integration Framework, is a uniform interface between the application and the transport. DMIF supports: a) An application interface irrespective of whether the peer is a remote interactive peer, broadcast or local storage media. b) Control of the establishment of a flexible way of interleaving packets of data with FlexMux channels c) the use of homogeneous networks between interactive peers: IP, ATM, mobile, PSTN, narrowband ISDN. d) the Support for mobile networks e) the use of user commands with acknowledgment messages f) the management of the synchronization of the information at the transport level

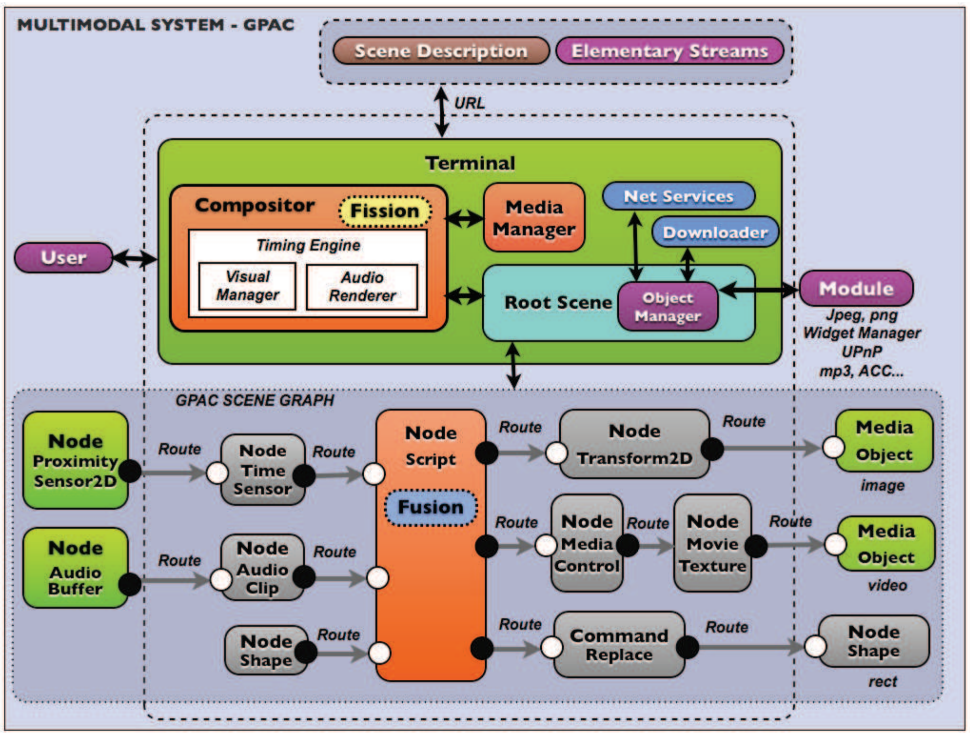


Figure 1.3.26: GPAC Multimodal Architecture

The Compositor responsibilities are:

- The object rasterization and presentation
- The mixing and rendering of audio
- The font management

In this way, every Terminal component can by default, handle data in *visual mode* and *auditive mode*, and supports interactive modalities and «rich» media with a fully time-oriented perspective mostly focused on the transport mechanisms (for its client-server nature). The type of support of every one of these features will depend on the User information and the available modules used by each instance.

To ensure this tasks the compositor contains four modules: a Timing Engine handling a Visual Manager and an Audio Renderer. [**Figure 1.3.26**]

The Compositor's Time Engine mdule, handles stream synchronization through an abstraction layer based on the MPEG-4 Synchronization Layer, allowing the application to drive media playback (play, pause, stop, fast- forward) and to perform dynamic stream switching -a kind of *data fission*- for example, for the selection of a specific media language while playing. This Time Engine can then support inter-language synchronization, while temporal information can be described using the MPEG model or the SMIL model [W3C-SMIL 1998].

The Compositor's Visual Manager is responsible of determining whether visual *modality* corresponds to the information fetched by its parent Compositor. For example, if it is a 3D graphics *modality* (3D scene), it handles the rasterization using OpenGL [SGI-OPENGL 1992].

Finally it returns the final media in the designated Scene (in turquoise) on the device. Thus, to accomplish its tasks the Visual Manager possess a Font Engine, a 2D graphics Engine and a 3D graphics Engine (capable to handle 3D universes and stereoscopic views) . [**Figure 1.3.26**]

The Compositor's Audio Renderer handles the audio output and the audio mixing. The audio input is managed by an Audio Input node.

The Audio Renderer handles the audio composition, a *data fusion* process by which decoded audio streams corresponding to different audio objects possibly encoded with different coding methods) are combined in the Terminal to form one or several audio tracks.

And it handles also the audio presentation, that is the processing related to playback the composed sound (that could be a spatial processed sound[62]) and the reproduction of the sound eventually with a *data fission* mechanism, via the corresponding output modalities (e.g. headphone, speaker or spatial sound structure). [**Figure 1.3.26**]

The second component of the Terminal is the Object Manager (in **violet**). It is used to link the content arriving by the elementary streams from the input Url to the Media Object abstraction. These elementary streams can be any visual or audio stream with commands or elementary streams conveying presentation and animation data, called the scene description which allows to attach time stamps and spatial information to such data.

The Object Manager (in **violet**) stores the input services selected according to the MIME type recognized by the Terminal. A service is an object providing data to the terminal for decoding and rendering, like a file reader or a socket reader. Possible input services in **GPAC** include broadcast file formats, AVI and MP3 files. Media data is passed to the associated Object Manager through channels.

With the networking services, the Object Manager handles also the extension modules needed for decoding. There are three types of decoder modules: video decoders, audio decoders and scene decoders. Each decoder module created, is associated to an Object Manager.

A Module (in **violet**) is an extension for the Terminal (external data needed to accomplish a task). In [**Figure 1.3.22**] the set of available modules data is represented by an unique component named Module. Nevertheless, at design time and at runtime, a GPAC player loads *n* modules corresponding to *n* decoders.

The Networking Services layer of **GPAC** is responsible for the management of the input services. It allows the player to support many file formats (locally or through HTTP), web radios, multicast and unicast, and a protocol stack for on-demand delivery of media streams. The player also features an MPEG-2 transport stream demultiplexer compatible with regular digital TV or mobile TV.

The third component of the Terminal is the Downloader component (in **blue**), used to download resources if needed by the Terminal. In the case of a downloaded module it is declared as a Terminal extension.

When object decoders are created by the Object Manager, they are registered with the fourth component of the Terminal: the Media Manager (in **green**). Depending on the decoder settings, a thread may be created to handle the decoder. Otherwise, the Media Manager, acting as a scheduler, will call each decoder running in non-threaded mode. [**Figure 1.3.26**]

The Root Scene (in **turquoise**) is an abstraction representing the space allocated to the restitution of media and its state. The Root Scene is populated with the streamed information about the presentation and its behavior. For example, when the scene decoder (loaded on the Object Manager) provides the composition instructions to generate a scene graph, this graph will populate the Root Scene.

These instructions are provided by the Scene Description (in **brown**), that is the functional core of the application, defining (with the BIFS language) the spatial and temporal position of the various objects, their dynamic behavior and the interactivity provided by the objects. It describes:

62 Sound that is carried out taking into account the relative positions of a sound source and a listening point.

- the audio-visual objects needed and their attributes (e.g. the Media Object Node in [**Figure 1.3.26**])

- the composition operations -in orange (e.g. with the Script Node in [**Figure 1.3.26**])

- the animation of the content (e.g. with the Transform2D Node in [**Figure 1.3.26**])

- the interactive behavior of individual objects -in green (e.g. with the Proximity-Sensor2D Node in [**Figure 1.3.26**]). The interactivity-related Nodes are called Sensors. To support input extensibility, a generic sensor is defined: the InputSensor Node.

The Scene Description contains also timing information that can be dynamically updated over time with the use of special Nodes called BIFS-Commands[63] (e.g. with the Command Node Replace in [**Figure 1.3.26**].

The Scene Graph generated from the Scene Description written in the BIFS language, provides an abstraction layer to the components responsible for input processing and output restitution of the data to the users via the Terminal device. The Scene Graph manages the behavior of the Root Scene and any modification and operation performed on it.

Like the Pipes and Filters architectural pattern, the Scene Graph consist of nodes that represent the objects connected by arcs (segments) that define relationships between nodes. Nevertheless, in the **GPAC** Scene Graph, an object is represented by either a node or a subgraph containing a group of nodes. This is a recursion mechanism, in which a Node in the Scene Graph can represent a nested Graph (called a subgraph). For example, in [**Figure 1.3.26**] the Transform2D Node contains a Graph with a Shape Node and its geometry containing a Text Node for the animated subtitles of a video that is played following a command triggered by a multimodal input (pointer and voice command).

Finally, to create a Terminal instance a User structure (in **violet**) is required [**Figure 1.3.26**]. This structure, associated to a configuration file, serves as a container for all information that is related to a user running the Terminal in opposition to the information stored in the Terminal object which is not specific to a user .

The goal of this module is to be able to use a same Terminal object with different users. It contains: a) settings for the modules to load, b)settings for the handling of the Terminal Window, c) settings to link an event processing function with OS related events occur (e.g. mouse move, quit ...) and d) a place holder for flags that indicates the user parameters for the Terminal: window-less mode, unthreaded terminal, no audio etc.

The multimodal characteristics of the **GPAC** multimedia player are showed in [**Table 1.3.49**] The player supports both directions, with an abstraction that represents the Device with the notion of Terminal. The three basic modes are supported with the use of dedicated nodes for *visual mode* and *the acoustic mode* and the possibility to extend input support by the use of the InputSensor Node.

Media is largely supported by the transport, coding and composition mechanisms, with dedicated nodes to handle signal analysis and mixing. [**Table 1.3.49**] Nevertheless a high level treatment of media can be allowed by the conformity with the MPEG-4 standard and its parts to handle the semantics of media (annotation, profiles, etc).

Fusion and *fission* are supported at the data, feature and modality levels, with dedicated functional blocks coming from the MPEG-4 standard (with a rich mechanism of synchronization) but also with the support of scripting with the Script Node as a way to extent the intelligent behavior of the multimodal or multimedia application. *Semantic fusion* or *semantic fission* are not explicitly supported even if the same Node could serve to add web semantics technologies to the interpretation and composition processes.

63 A single change to the Scene. Changing the position of an Object may be done with a BIFS-Command. The basic commands are insert, Delete and Replace.

Direction	Both	*YES*	
Level of Abstraction	Input Abstraction	Device	*TERMINAL*
	Output Abstraction	Device	*TERMINAL*
Supported Modes	Acoustic	*YES*	
	Visual		
	Haptic		
	Other	*NO*	
Multiple Modalities for each Supported Mode ?	*YES*		
Media Support	*YES*		
Fusion	Data	*YES*	
	Feature	*YES*	
	Semantic	*NO*	
	Hybrid	*NO*	
Fission	Data	*YES*	
	Modality	*YES*	
	Semantic	*NO*	

Table 1.3.49: Multimodal Characteristics in GPAC.

The multimodal session as defined in our criteria is supported at different levels. [**Table 1.3.50**] To manage the streaming of contents, the MPEG-4 standard defines a notion of session, which is multimodal by its audiovisual nature. Secondly, the Terminal instances implement a mechanism of authentication and configuration by user, with information about the loaded extensions (decoders) which is at least, a general description about the modes and modalities that the Terminal can support. Finally, some components like the Object Manager, the Scene Graph and the Visual Manager provide information about the resources allocated to multimodal input /output depending on the Scene Description.

Session	Supported	*YES*
	Handle by Manager	*YES*
	Migrated	*YES*
	Historized	*YES*

Event	Supported	*YES*	Temporal Order	*YES*
	Handle by Manager	*YES*	Monomodal Support	*YES*
	Synchronized	*YES*	Multimodal Support	*NO*
	Disambiguated	*YES*	Associated to context	*NO*
	Multiple Granularity	*YES*		

Interaction (Dialog)	Multimodality	Sequential	*YES*	Interaction Types	Direct	*YES*
		Simultaneous	*YES*		Free flow	*NO*
		Composite	*NO*			
	Synchronized	*YES*		Focus Handled	*YES*	
	Historized	*NO*		State recorded	*YES*	
	Turn Taking	*YES*		Context updated	*NO*	
	Strategies	One	*YES*	Interpretation	by grammar	*YES*
					by structure	*YES*
		Multiple	*NO*		by planning	*NO*
					by learning	*NO*
					by intention int.	*NO*

Table 1.3.50: Management of Multimodal Behavior in GPAC.

The session migration in GPAC

One example of implementation to handle a possible multimodal session, using these components and a dedicated manager (as an extension) is the implementation of widgets migration in **GPAC**. [Le Feuvre et al. 2009]

With this proposal of distributed widgets, we can see a concrete case of session handling at the applicative level, supported by a Widget Manager module, to allow widgets to be transferred between Terminals keeping a history of its state. [**Table 1.3.50**]

Events are supported in an extensive manner as showed by [**Table 1.3.50**]. Events are treated at multiple granularities, from the signal level to the user interaction level, and application level.

Nevertheless, they are not triggered by multimodal processes and they are not associated to an extended notion of context.

Direct interaction is supported by the use of two strategies, the state machines (grammars) that are usually implemented with the script node and the event handling with DOM and the Scene Graph structure. [**Table 1.3.50**]

Both strategies ensures simultaneous and sequential interaction, turn taking, focus handling and modality synchronization. The state of the interaction is carried by nodes and managed by the Root Scene.

Concerning the participants in the multimodal system, Devices are not explicitly modeled but they are described by profiles with the Terminal abstraction. This profile is defined by a configuration document, and by the BIFS API. [**Table 1.3.51**]

Users are described by a profile with a focus on personalization (terminal parameters and configuration) and some session handling without a data collection or inference mechanisms. This profile is distributed among the Terminal instances. [**Table 1.3.51**]

Device Modeling	**At Model Level**		*NO*	**Abstract Description**		*YES*
	At Profile Level		*YES*	**Multimodal Info**		*NO*
	Managed	by Document	*YES*			
		by API	*YES*			
		by Service	*NO*			
		by Repository	*NO*			
User Modeling	**At Model Level**		*NO*	**Data collection**		*NO*
	At Profile Level		*YES*	**Personalization**		*YES*
	Generation	Deducted	*YES*	**Levels**	Sensorial	*YES*
		Inferred	*NO*		Behavioral	*YES*
					Semantic	*NO*
	Stored	Centralized	*NO*	**Stereotypes**		*NO*
		Distributed	*YES*	**Social support**		*NO*
Domain Modeling	**Application**		*NO*	**Interaction**		*YES*
	Presentation		*YES*	**Use**		*NO*
	Life-cycle	Design	*YES*	**Granularity**	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*NO*
	Interaction Classification	Categories	*NO*	**Abstract**		*NO*
				Taxonomies		*YES*
		Design Spaces	*NO*	**Performance Semantics**		*NO*
		Temp. Relations	*NO*			

Table 1.3.51: Participants in GPAC.

The interaction domain is modeled at the presentation level and guided by the sensor proposal of BIFS. TheTerminal behavior is described for all the life-cycle with a granularity related to actions and sequences of actions but not with a high level interpretation of the action mean. The BIFS nodes contribute with an interaction taxonomy based on the MPEG-4 standard and its origins on virtual reality issues. [**Table 1.3.51**] The interaction classification that can guide designers for the use of specialized primitives is not addressed.

Finally, the environment and the context are covered from the temporal and spatio-temporal point of view.

Time is handled as points and as intervals, but the semi-interval approach is not tested. The synchronization mechanism uses valid time described for each node and the transaction time needed by the player to handle streams. Valid time is also described by the Scene Description for the pipeline management and the dynamic updates of the Root Scene. As a media player, **GPAC** handles the homogeneous distribution of data arriving in the elementary streams, and as a rich media player it handles heterogeneous data produced by the system behavior or the user behavior. [**Table 1.3.52**]

The space-time situation is represented in the generic Root Scene, used to contain the spatial and temporal information coming from the Scene Description. Thee Scene notion is by default a spatial notion in which geometric and symbolic coordinates support all types of spatial relations and point of view (allocentric mostly for 2D features and egocentric for 3D features). No social or emotional bias are proposed for the presentation or behavior of the Scene. [**Table 1.3.52**]

Temporal Situation	**As Points**		*YES*	**Valid Time**		*YES*
	As Intervals		*YES*	**Transaction Time**		*YES*
	As semi-Intervals		*NO*	**Symbolic Time**		*NO*
	Sensed Data Distribution	Homogeneous	*YES*	**Returned Data Distribution**	Homogeneous	*YES*
		Heterogenous	*YES*		Heterogenous	*YES*
	Relational Aspects		*NO*	**Temporal Operators**		*NO*
Space-Time Situation	**Ontologies**		*NO*	**Emotional Bias**		*NO*
	Other Context Descriptions		*NO*	**Social Bias**		*NO*
	Coordinates	Geometric	*YES*	**Proxemic**	Distance	*NO*
		Symbolic	*YES*		Dynamics	*NO*
	Relations	Metrical	*YES*	**Qualitative Relations**	Quantity Spaces	*YES*
		Ordinal	*YES*		Cardinal	*YES*
		Topological	*YES*		Grid-Based	*YES*
	Views	Allocentric	*YES*	**Naïve Operators**		*NO*
		Egocentric	*YES*			

Table 1.3.52: Environment and context in GPAC.

The Contribution of Gpac for a Multimodal Architecture of Reference

To sum up, we can point out that the first contribution of the **GPAC** player from the perspective of a Multimodal Architecture of Reference is its very detailed treatment of spatial and temporal data for presentation and synchronization of medias. The update mechanism inherited of the MPEG standard is a starting point for our proposal to handle updates in our architecture.

The second contribution could be its approach of command and content data as dynamic streams (useful for real-time use cases) and services to be consumed by the application (useful for a service oriented perspective of multimodal systems).

The third contribution is the rich support of 3D modalities reflected by the use of interaction nodes that are defined in spatial terms. This can be useful for multimodal systems needing context-awareness strongly related to the space properties.

Finally, the fourth contribution is the management of the session that can enrich the properties needed for a multimodal session in cases of modality migration, for example, to adapt the behavior of the application to the user needs (e.g. in mobile use cases).

1.3.3.6 ELOQUENCE - 2004

ELOQUENCE [Rousseau et al. 2004] is a authoring framework for the design of multimodal applications focused on the multimodal presentation (output) of the information. This software platform includes a set of tools that has been tested in the implementation of a cockpit simulator.

The conceptual model of Eloquence

The architecture of **ELOQUENCE** applies a conceptual model called WWHT (the What, Which, How and Then model). The model views the design process as a series of steps. The first step -What, is made during the specification process of information units to present. The three other steps -Which, How and Then- are managed by different modules of the architecture model.

The model proposes a contextualization of the multimodal presentation thanks to a decisional process based on election rules. It also handles evolution of the presentation through the introduction of two different approaches to ensure coherence and validity in regard to the evolution of the context.

The **ELOQUENCE** platform incarnates this design cycle model composed of three steps: analysis, specification and simulation. It helps the designer by proposing a set of tools allowing the specification, simulation and execution of the system outputs.

The architecture model is composed of four main modules: the Dialog Controller (in orange), the Election Module (in pink), the Instantiation Module (in orange), the Multimodal Presentation Management Module (in orange) and the Context Spy Module (in turquoise). [**Figure** 1.3.27] The knowledge of the system is defined through five kind of data structures (in **violet**): context models (three models and seven criteria), behavior rules (eleven election rules), contents list, attributes instantiation models and a multimodal presentation list. [**Figure** 1.3.27]

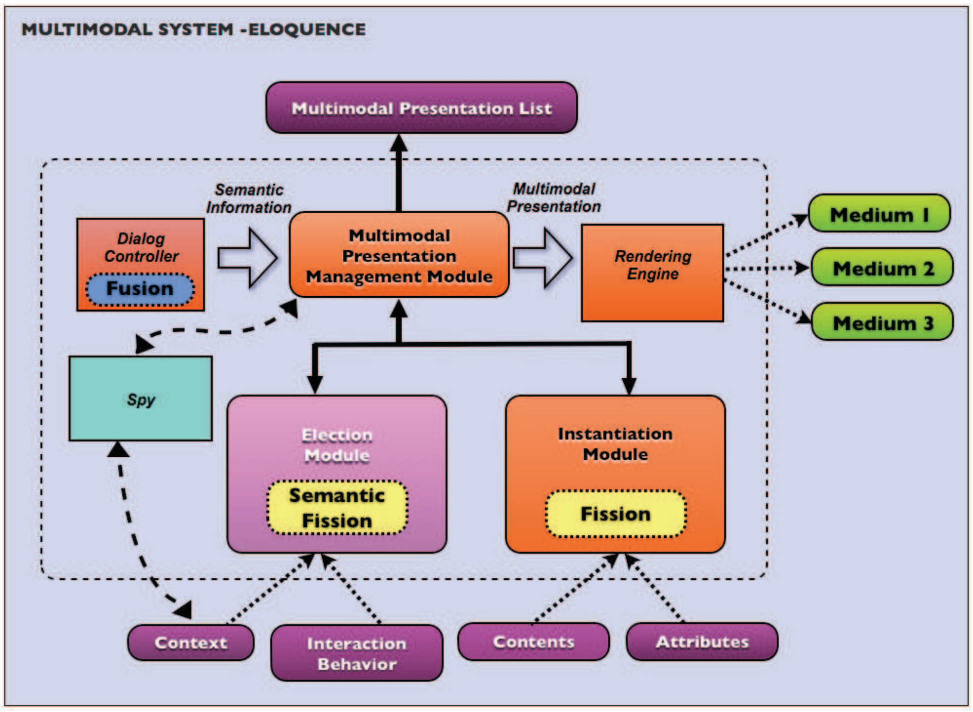


Figure 1.3.27: ELOQUENCE Multimodal Architecture

The Dialog Controller (in orange) [**Figure** 1.3.27] communicates with the Multimodal Presentation Management Module (in orange) through messages. The Multimodal Presentation Management Module treats the messages coming from the Dialogue Controller by generating a multimodal presentation that expresses the semantic contents associated with the message. Once the allocation of a multimodal presentation is finished, this presentation is transmitted to the different Media (in green).

In the **ELOQUENCE** platform, the Election Module (in **pink**) is the center of the architecture. It builds the multimodal presentation by using a knowledge data distributed in models of the interaction context and rules defining the election behavior. The Election Module applies rules that modify the contextual weights associated to the interaction components (at the *mode*, *modality* and *media* levels) managed by the system.

The values of these contextual weights depend on the interaction context and will guide the decision of which multimodal presentation is selected. Once the rules are applied the best *modality-medium* pair is selected. The system will then make the choice to enrich or not the multimodal presentation. To enrich a presentation means to select new modality-medium pairs using the CARE properties[64] (e.g. redundant or complementary pairs). Finally, the Election Module sends to the Multimodal Presentation Management Module the selected multimodal presentation.

The Multimodal Presentation Management Module (in **orange**) centralizes the communications of the architecture's components. It distributes the work load between the components and manages a list of multimodal presentations currently actives: the Multimodal Presentations List. [**Figure 1.3.27**]

This data module stores a registry of the active multimodal presentations and pointers to the context elements making these presentations valid. In this way, the Multimodal Presentation Management Module is responsible to check the validity of the active multimodal presentations.

To accomplish this task, it receives information from the Spy Module (in **turquoise**) that analyzes the evolution of the interaction context and announces any modifications of an element of the context having any influence on a multimodal presentation currently active. [**Figure 1.3.27**]

Thus, the Multimodal Presentation Management Module can also decide to cancel the validity of some presentations. In the case of a cancellation of an active multimodal presentation, the it request the Election Module to launch a new selection. The Election module will then return a multimodal presentation associated with a context criteria in order to ensure the validity of the new presentation. [**Figure 1.3.27**]

So, the Spy Module can be the source of the invalidation of an active multimodal presentation. As soon as a multimodal presentation is selected and added to the Multimodal Presentation Management Module , the Spy Module receives a list of criteria of the interaction context from which this presentation depends. The Spy Module also supervises the modification of these criteria.

The **ELOQUENCE** platform, proposes a distinction between the choice of modalities and the way they will be instantiated (the interaction performance). The modalities choice is done by the Election Module while the Instantiation Module determine how to present the information through these modalities by generating the multimodal presentation allocated by the Election Engine.

Therefore, the Election Module decides "Which modalities to present the information?"(the representation level) whereas Instantiation Module decides "How to present the information using these modalities?" (interpretation/performance level).

At the representation level, the content is chosen to express a semantic message through the modalities composing the allocated multimodal presentation. At the performance level, choices are made on the presentation parameters (position, modalities attributes, etc.) to better communicate this message according to a context of interaction.

One example given by the authors is a call notification. The Election Module decides to express the information through a multimodal presentation composed of: a text *modality* with 2D Graphics *mediu*m and (using redundancy) a ring sound *modality* with the phone speaker *medium*.

At the representation level an output text "call of ..." is chosen for the first *medium* and the pink panther song for the second *medium*. At the performance level it is decided to put text in the center of the screen with a significant character font for the first *modality* and to turn the volume up for the second.

64 See Supra 1.2.2.3 Domain Modeling

Finally, the **ELOQUENCE** platform details the procedure to determine the rules needed to support the selection mechanism. A rule used by the Election Module consists of two parts: a set of premises (preconditions) and a set of conclusions (post conditions).

The premises are based on the knowledge stored in seven models (in **violet**): the model of interaction, the task model, the information model, the environment model, the user model, the dialog model and the system model. The conclusions concern five types of objects: the *mode*, the *modality*, the *medium*, a taxonomical criteria and the CARE properties.

When a conclusion is made about a *mode*, a *modality* or a *medium* object, it has the effect of changing the weight or the contextual status of this object.

When a conclusion derives from a taxonomical criterion, it changes the status of a set of objects (modalities or media) according to the nature of this taxonomic criteria.

Finally, a conclusion on a CARE property object indicates the need to enrich the multimodal presentation by performing one or multiple selections according to the complementarity or redundancy properties.

To handle the multimodal characteristics [**Table 1.3.53**], **ELOQUENCE** points out only the output direction with an abstraction for the output side, viewing devices as media. Three modes are supported with multiple modalities. An extended theoretical support of media is proposed (for the design process) by the analysis of the context of interaction in terms of the relations between *mode*, *modality* and *medium*.

The *fusion* is supported by the Dialog controller at the feature level, but not described with details in **ELOQUENCE**. In contrast, *fission* is supported at the modality level by the Instantiation Module and at the semantic level by the Election Module. [**Table 1.3.53**]

Direction	Output	*YES*

Level of Abstraction	Output Abstraction	Device	*MEDIUM*

Supported Modes	Acoustic	*YES*
	Visual	*YES*
	Haptic	*YES*
	Others	*NO*

Multiple Modalities for each Supported Mode ?	*YES*
Media Support	*YES*

Fusion	Data	*NO*
	Feature	*YES*
	Semantic	*NO*
	Hybrid	*NO*

Fission	Data	*NO*
	Modality	*YES*
	Semantic	*YES*

Table 1.3.53: Multimodal Characteristics in ELOQUENCE.

In **ELOQUENCE**, multimodal sessions or events are not described, but the interaction and decision are covered, the first with the exchanges between a Dialog Controller and a Multimodal Presentation Management Module, the second is distributed between the Election Module and the Instantiation Module. [**Table 1.3.54**]

Concerning the Interaction, the Multimodal Presentation Management Module handle sequential and simultaneous multimodal presentations, synchronizing them based on the updates of the context provided by the Spy Module. This module is responsible to historize the state of the interaction in order to compare this information with the information provided by the models and more specifically by the Interaction Context model. [**Table 1.3.54**]

For planning the interaction, multiple strategies are proposed, with the use of rules that can be used with finite-state, frames or information-state strategies. In contrast, the learning and the interpretation of the user intentions is not addressed. [**Table 1.3.54**]

Interaction (Dialog)	Multimodality	Sequential	*YES*	Interaction Types	Direct	*NO*
		Simultaneous	*YES*		Free flow	
		Composite	*NO*			
	Synchronized		*YES*	Focus Handled		*NO*
	Historized		*YES*	State recorded		*YES*
	Turn Taking		*NO*	Context updated		*YES*
	Strategies	One	*NO*	Interpretation	by grammar	*YES*
					by structure	*YES*
		Multiple	*YES*		by planning	*YES*
					by learning	*NO*
					by intention int.	*NO*
Decision	Goal	Satisfy constraints	*YES*	Techniques	Formal	*YES*
		Reduce options	*YES*		Non Formal	*NO*
		Recommend	*NO*			
	Knowledge	System	*YES*	Uses	Resolve ambiguity	*YES*
					Uncertain reasoning	*NO*
		Domain			Support interpretation	*NO*
					Collaborate in planning	*YES*
		General	*NO*		Support RW understanding	*NO*

Table 1.3.54: Management of Multimodal Behavior in ELOQUENCE.

The Election Module centralizes the semantic decision about which modalities must be used to present the information. This decision process is executed to satisfy constraints given in the formal rules and models, and to reduce options, but no recommendation mechanism is described. In this way, decision can help to resolve ambiguity and collaborate in planning, but in **ELOQUENCE** it is not extended to help in uncertain reasoning (e.g. for mobile applications), neither to support input interpretation or real world understanding. Finally, the decision process is based on the knowledge sources of the system and the domain and this process enriches this models with the activity of the Spy Module. [**Table 1.3.54**]

User Modeling	At Model Level		*YES*	Data collection		*NO*
	At Profile Level		*YES*	Personalization		*YES*
	Generation	Deducted	*YES*	Levels	Sensorial	*YES*
		Inferred	*NO*		Behavioral	*YES*
					Semantic	*YES*
	Stored	Centralized	*YES*	Stereotypes		*NO*
		Distributed	*NO*	Social support		*NO*
Domain Modeling	Application		*YES*	Interaction		*YES*
	Presentation		*YES*	Use		*YES*
	Life-cycle	Design	*YES*	Granularity	Action	*YES*
		Load	*YES*		Sequence	*YES*
		Run	*YES*		Mean	*YES*
	Interaction Classification	Categories	*YES*	Abstract		*YES*
				Taxonomies		*YES*
		Design Spaces	*YES*	Performance Semantics		*YES*
		Temp. Relations	*YES*			

Table 1.3.55: Participants in ELOQUENCE.

The participants of the multimodal system are covered from the user and domain models. The user is modeled with two structures: the dialog model and the user model. The user model is described by two sets of attributes, the static attributes and the dynamic attributes. [**Table 1.3.55**]

Some static attributes are the physical ability, language of the user, the user goals, and the preferences for the application and for the interaction. Dynamic attributes correspond to the level of experience for a given task (e.g. navigation), the error rate, the user disorientation. [**Table 1.3.55**] This attributes show how the user modeling is covered at the model and profile levels to reflect sensorial, behavioral and semantic data in a centralized structure. However, this modeling mechanism is not intended to collect user data

As show in [**Table 1.3.55**] the criteria concerning the domain data is totally covered, and this is possible because it is modeled in detail in the interaction context model. For example, by the use of the CARE properties, design spaces and temporal relations between modalities are covered. A taxonomy of modes, modalities and media is proposed and described from an abstract point of view. An emphasis is given to the semantics of the performance of modalities, with the use of a Instantiation Module and the study about the difference between the modalities selection and their instantiation. As a design tool, ELOQUENCE addresses the entire lifecycle of the multimodal application, with a special focus on the modeling of the interaction and the presentation of the information. [**Table 1.3.55**]

The environment and context are not viewed in ELOQUENCE as we defined for our criteria. For the authors a context of interaction is the combination of modalities given a precise user and a precise state of the system. In contrast for our criteria, the environment and context are the temporal and spatio-temporal conditions in which the interaction occurs. These conditions are not covered in ELOQUENCE.

Nevertheless, the usage situation is addressed with the model covering the interaction context, at the level of a model but also as a profile level. This situation is described as the relationship between modes, modalities and media within the context of interaction model.

Usage Situation	**At Model Level**	*YES*	**Relationships**	*NO*
	At Profile Level	*YES*	**Mental Models**	*NO*
	Factual Description	*NO*	**Social Qualities**	*NO*
	Interpretation Description	*NO*		

Table 1.3.48: Environment and context in ELOQUENCE.

In short, the ELOQUENCE's main contributions for a Multimodal Architecture of Reference can be the focus on the instantiation notion (performance/interpretation level) with a dedicated component; the proposal of a component, the Spy Module, to monitor the state of the interaction cycle; and the proposal for the definition of rules based on premises/conclusion, information units and multimodal taxonomies.

The Contribution of Eloquence for a Multimodal Architecture of Reference

1.3.3.7 HEPHAISTK - 2008

HEPHAISTK [Dumas 2010] is an agent-based framework for the creation of multimodal interfaces mixing tangible interaction with other forms of input and a practical tool for the study of a canonical multimodal architecture proposed by the author.

HEPHAISTK offers a predefined set of recognizers as well as the possibility to plug into the toolkit any other modality recognizer that supports communication using the EMMA language [W3C-EMMA 2009]. It aims at providing a tool allowing developers to easily prototype multimodal interfaces by plugging it in a client application that wishes to receive notifications of multimodal events received from a set of modality recognizers.

In this way, the client application delegates to **HEPHAISTK** the management of the multimodal interaction. Therefore, it is necessary to monitor that the toolkit and the application are in the same state. [**Figure 1.3.28**]

Every recognizer agent (in green) is asked to specify its corresponding connector. The main task of a recognizer agent is to encapsulate and forward all input data generated by the external recognizer, as well as metadata regarding this data. To manage a given external recognizer (in green), **HEPHAISTK** uses the recognizer agents as connectors. A connector (in green) is at the same time, an abstract representation of a given class of input data, and a provider of every facility needed to manage the transmission of these input data to the framework.

An external input recognizer focuses on one given modality source, such as speech, gesture, emotions... or one particular device, such as the mouse, or a multi-touch table. It can be any type of software, as long as they are able to transmit their results to code written in the Java language.

The **HEPHAISTK** agents, collaborate through a blackboard, and manage individual modality recognizers, handle *fusion* and manage the dialog. The tool can be configured with the SMUIML language (Synchronized Multimodal User Interfaces Markup Language) used to describe the human-machine multimodal dialog and to control the multiple input modalities to be fused. [**Figure** 1.3.28]

In the central blackboard architecture of **HEPHAISTK**, all data coming from the different sources are standardized in a central place, where other interested agents can dig them at will. This blackboard, called the Postman (in violet), centralizes each message coming from the different input recognizers and stores it into a database. The agents interested in a specific type of message can subscribe to the Postman, which will accordingly redistribute the received messages. [**Figure** 1.3.28] This mechanism of subscription allows for example, the dynamic modifications or the change of the method of *fusion*. The Postman also manages timestamps and handles the normalization between data coming from different input sources. Nevertheless, this central agent does not act like a facilitator because only agents dealing with recognizers-wise input are able to send data to it.

The Integration Committee (in orange) is responsible of the extraction of meaning from data coming from the different input recognizers. [**Figure** 1.3.28] This Integration Committee is instantiated for a given client application by a SMUIML document (in violet). The SMUIML document contains information about the dialog states in the Dialog Manager; the events leading from one state to another in the *Fusion* Manager (in blue -dotted); and the information communicated to the client application, given the current dialog state and context in the *Fission* Manager (in yellow -dotted).

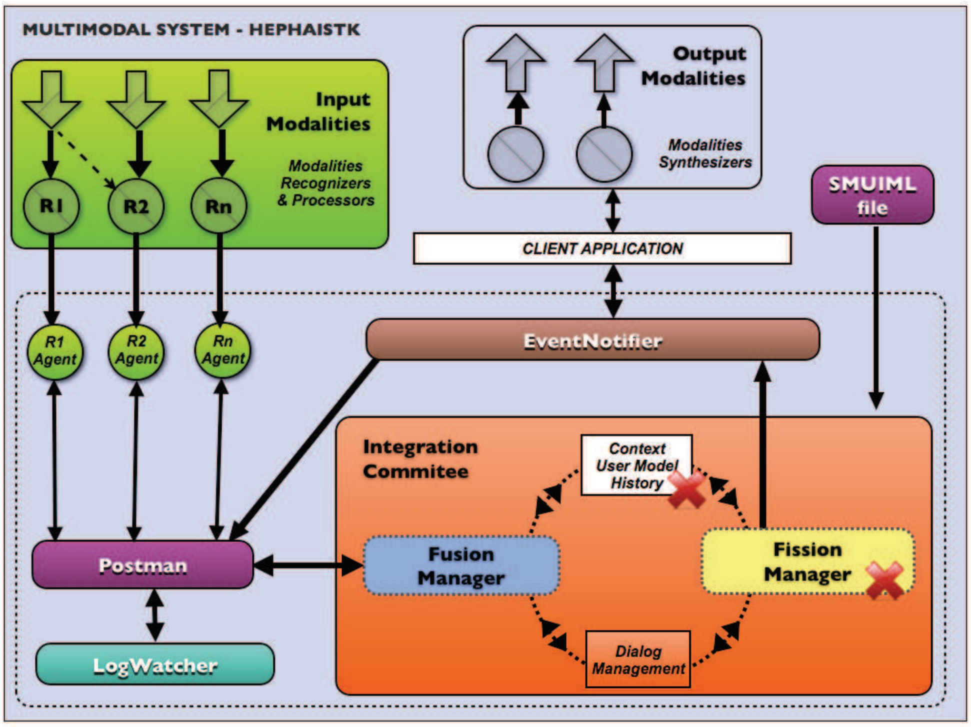


Figure 1.3.28: HEPHAISTK Multimodal Architecture

The Fusion Manager agent receives from the Postman new input events and feeds them to the fusion algorithm that is currently selected, which, would trigger a result, in the form of a message to be sent to the client application, indicating the user interaction.

In **HEPHAISTK** the *Fission* Manager is only a dummy agent, forwarding data without any processing due to the lack of management of context and user profiles.

When a developer wants to monitor input, he has to declare the **HEPHAISTK** toolkit using event listeners. The *Fusion* Manager and the Dialog Manager (in **orange**) are scripted using a SMUIML file[65] which declares recognizers, triggers and actions, and the user-machine dialog in the form of a finite state machine calling those triggers and actions.

While the general dialog scheme is fixed, the behavior of the *fusion* engine can be adapted by the developer to match the different CARE properties[66], for example, by specifying how modalities will be fused, in a parallel or complementary way.

The Integration Committee sends all *fusion* results to an Event Notifier agent, responsible to interface the client application with the framework: it is used by the client application to communicate with the framework and to provide higher-level information.

As a result, **HEPHAISTK** sees the client application as its client, but also as another input source, and consequently the Event Notifier takes also the role of a recognition agent which communicates through a set of messages predefined in the SMUIML script.

Finally, the Log Watcher agent (in **turquoise**) is responsible to log events from the Postman agent and to output them to in a file or a GUI window, following defined rules. It handles the state of the interaction and logs it.

As [**Table 1.3.57**] shows, the **HEPHAISTK** toolkit only supports Inputs which are treated with a high-level abstraction as *Sensor Systems*. Three modes are supported focusing on the recognizing tasks. *Fission* is not supported while *fusion* is only described at the feature level.

Direction	Input	*YES*	
Level of Abstraction	Output Abstraction	System	*INPUT RECOGNIZER*

Supported Modes	Acoustic	*YES*
	Visual	*YES*
	Haptic	*YES*
	Others	*NO*

Multiple Modalities for each Supported Mode ?	*YES*
Media Support	*NO*

Fusion	Data	*NO*
	Feature	*YES*
	Semantic	*NO*
	Hybrid	*NO*

Fission	Data	*NO*
	Modality	*NO*
	Semantic	*NO*

Table 1.3.57: Multimodal Characteristics in HEPHAISTK.

Concerning the management of the multimodal behavior, the multimodal session or the decision issues are not addressed with **HEPHAISTK**, whereas events are handled by a specific manager, the Event Manager, and synchronized and disambiguated in the Postman. These events are not described as carrying multimodal information but rather only of monomodal information. The temporal order of events is also ensured, event if events are not associated to a mechanism for updating a model of the context of interaction. [**Table 1.3.58**]

65 The description of this language is out of the scope of our work.
66 See Supra 1.2.2.3 Domain Modeling

Event	Supported		*YES*	Temporal Order		*YES*
	Handle by Manager		*YES*	Monomodal Support		*YES*
	Synchronized		*YES*	Multimodal Support		*NO*
	Disambiguated		*YES*	Associated to context		*NO*
	Multiple Granularity		*NO*			
Interaction (Dialog)	Multimodality	Sequential	*YES*	Interaction Types	Direct	*YES*
		Simultaneous	*YES*		Free flow	*NO*
		Composite	*NO*			
	Synchronized	*YES*		Focus Handled	*YES*	
	Historized	*YES*		State recorded	*YES*	
	Turn Taking	*YES*		Context updated	*NO*	
	Strategies	One	*NO*	Interpretation	by grammar	*YES*
					by structure	*YES*
		Multiple	*YES*		by planning	*YES*
					by learning	*NO*
					by intention int.	*NO*

Table 1.3.58: Management of Multimodal Behavior in the HEPHAISTK.

The interaction is handled sequential and simultaneously to cover direct interaction by the means of multiple strategies: finite state machines and Hidden Markov Models. These strategies collaborate on planning the interaction, the turn taking mechanism, the handling of the focus of the interaction and the recording of the state. Nevertheless, there are no learning support or interpretation of the user intention. [**Table 1.3.58**]

Devices are modeled by the view of input recognizers as connectors. This allows to model devices with an dedicated API and a description file (written in the SMUIML language). Yet, any profile information is proposed, containing an abstract description of the device attributes or any multimodal informations. [**Table 1.3.59**]

Device Modeling	At Model Level		*YES*	Abstract Description		*NO*
	At Profile Level		*NO*	Multimodal Info		*NO*
	Managed	by Document	*YES*			
		by API	*YES*			
		by Service	*NO*			
		by Repository	*NO*			
Domain Modeling	Application		*YES*	Interaction		*YES*
	Presentation		*NO*	Use		*NO*
	Life-cycle	Design	*YES*	Granularity	Action	*YES*
		Load	*NO*		Sequence	*YES*
		Run	*YES*		Mean	*YES*
	Interaction Classification	Categories	*NO*	Abstract		*YES*
				Taxonomies		*NO*
		Design Spaces	*YES*	Performance Semantics		*NO*
		Temp. Relations	*YES*			

Table 1.3.59: Participants in HEPHAISTK.

The interaction domain is viewed from the perspective of an application delegating the management of the multimodal interaction to the toolkit. This is described for the phase of design and run of the application.

As an input framework, the presentation is not covered. The use is not described, and neither taxonomies nor the semantics of the performance of modalities or the categories of interaction. In contrast, the framework uses the CARE design spaces and the temporal relations expressed with its properties. [**Table 1.3.59**]

The HEPHAISTK toolkit does not cover the information about user, the environment or the temporal and spatio-temporal context for the multimodal interaction.

The Contribution of HephaisTK for a Multimodal Architecture of Reference

To finish, **HEPHAISTK** can contribute to a Multimodal Architecture of reference in some points. First, its approach of the client application as an entity delegating the management of inputs to a framework with the use of a well-designed interface. Secondly, the proposal of a dedicated component to handle the multimodal events. Finally, the definition of a blackboard-like Postman, as a messages dispatcher.

1.3.4 Conclusion

The sample of sixteen architectures presented in this section was selected from a previous analysis of a larger set - one hundred - of multimodal implementations[67]. The selection criteria has been the amount of information provided by the authors about the architectural facets of the implementation, its completeness and its representativeness of some domains of research that we found relevants.

During this section three levels of analysis were provided: first, a functional description of the architecture, second, a visual comparison of the functional blocks expressed with color codes and finally, the analysis of four sets of criteria gathering seventeen characteristics :

- Characteristics related to the elementary description of the multimodal system; e.g. direction, number of modes supported, abstraction level for the inputs/outputs, type of *fusion/fission*.
- Characteristics related to the system's and user's behavior regarding the session management, the event handling, the dialog strategy or the decision making process.
- Characteristics related to the description of some interaction participants : devices, users and domain data.
- Characteristics related to the general delivery context of the system: the usage situation, the temporal situation and the spatial situation.

For each architecture studied, a few number of interesting facts were highlighted with the perspective of the enrichment of an architecture of reference envisioned for the purpose of standardization and interoperability.

As a result, the following section will present two groups of trends that emerge from the choice of functional blocks of our historical sample of multimodal architectures. An emerging trend is a topic area for which one can trace the growth of interest and utility over time.

By analyzing the selected contributions, we can detect a non-disruptive emergence of topics that build on previously existing topics treated by the multimodal research community.

These are novel trends that are globally derived from the multimodal interaction domain, but that provides new challenges to the architectural decisions for the implementation of multimodal systems.

In the following sections we will present a first group of emerging trends directly related to the criteria and a second group, transversal to all criteria.

67 See Appendix 1 Multimodal Architectures Timeline

1.3.4.1 Emerging trends directly related with the criteria

We can identify emerging trends directly related with the four sets of characteristics listed above.

The first recurrent topic is the event handling and corresponds to the second set of studied characteristics: the system's and user's behavior. Seven architectures of our sample try to address the management of events, which is normal in the human computer interaction research because user interfaces are highly event-oriented.

Nevertheless this is an aspect even more striking in multimodal interfaces, given the superposition of input and output layers and the use of techniques like the redundancy, the complementarity or the equivalence.

The concerns about the event management in our selection, are resolved with triggers categories (OAA), macrocommands (GALATEA), task control layers (OPENINTERFACE), transport queues (MEDITOR), hardwired mechanisms (REA), scripted handling (DIRECTOR) and dedicated components (HEPHAISTK).

The MMI Framework & Architecture respond to the same concern with the Interaction Life-Cycle Events, and the proposal of a dedicated component: the Interaction Manager. It also provides a clear separation between the interaction control data (with the Life-Cycle Events) and the interaction content data.

In contrast, hardwired mechanisms are not envisioned for the moment, neither the transport queue mechanism implemented in MEDITOR, GPAC and HEPHAISTK as an important phase for the *fusion / fission* of modalities. In consequence, these can be possible extensions to the architecture of reference for providing some complementary resources to handle the multimodal events.

The second key topic, recovered from five of the sampled architectures is the state management. It corresponds also to the system's and user's behavior characteristics listed above.

This feature is oriented to register the evolution of the interaction cycle and provide the information about any modification of the state of the system and the components. It is designed as a monitoring process (ELOQUENCE, SMARTKOM, HEPHAISTK) in support of the decision layer, as a display list manager (DIRECTOR) in support of the *fusion* and *fission* mechanisms, as a blackboard (OAA, HEPHAISTK), as an object manager used for decoding and rendering purposes (GPAC), and as a routing table (MEDITOR).

Concerning this subject the MMI Framework recommends a specific component to handle the multimodal session and the state of components [**Figure 1.3.9**]. Yet, it does not give details about the interfaces needed to use this component or about its role in the management of the interaction cycles. As a result an extension to the MMI Framework & Architecture can be conceived to complete this generic description with specific details about the eventual implementation, behavior and responsibilities of this Session component.

The third topic that we can raise from our sample of architectures is the need of a generic method to handle the decision functional block and it is also related to the second set: the system's and user's behavior.

Two approaches are used to explore this subject: a knowledge-oriented approach of decision (MEDITOR, FAME, REA, ELOQUENCE) and an action-oriented approach of decision (OAA, SMARTKOM).

In both approaches a dedicated component is defined, and we must highlight a key proposal from REA and SMARTKOM who implements a shortcut strategy to handle the interaction (inputs/outputs) avoiding the decisional process and as a result, saving time. This is the goal of the direct links between Modality Objects (in **green**) and the Function Modeling component (in **brown**) in SMARTKOM [**Figure 1.3.25**] and the goal of the Harwired Reactions (in **violet**) in REA [**Figure 1.3.23**].

Another important contribution from REA to the definition of a functional block for decision handling, is the proposal of an Understanding Module (in **blue navy**) [**Figure 1.3.23**], relying a preliminary knowledge processing to the decisional mechanism.

For its part, the MMI Framework & Architecture does not address decision processing or generation. However, the implementation of these six architectures show the relevance and common need for the decision support in multiple kinds of *multimodal systems*: Embodied Conversational Agents, Multimodal Presentation, Mobile Applications, Fight Simulation and Ambient Intelligence.

Such a component is a precious tool to help to resolve ambiguity and to collaborate in planning, but also in more complex use cases in which is necessary to support the rich interpretation of inputs, to reason in uncertain and dynamic conditions and to understand real world conditions to better adapt the system to the situation of interaction with a user.

Hence, we can foresight another extension to the MMI Framework & Architecture adding an interface to a dedicated component for decision, that can be implemented in a simple or complex way depending on the application needs. This component can also be relied to some knowledge process and can collaborate to a standardized mechanism to avoid decision making for automatic response or automatic input needs[68] or for the direct control of modalities by a client application (when the multimodal management is delegated to the system by an external client application[69]).

Finally, the fourth topic emerging from our sample of multimodal architectures is the inclusion of social behavior to the implementation of multimodal systems.

This topic is revealed by the use of emotional information to enrich the user and the interaction models, the system's output behavior and to control the *fusion* and the *fission* processes (REA), the support of cultural information for the interpretation of inputs and the composition of outputs (GALATEA) and finally, the intention approach that treats the user's plans to achieve desired effects on the mental states of the participants in the interaction (SMARTKOM).

These intentions are defined in an ontology and it can be used to interpret even an ironic comment or to understand the user's dissatisfaction, which are mental states deeply marked by social and environmental conditions.

The MMI Framework & Architecture respond to this concern with a generic component dedicated to the System and Environment management [**Figure 1.3.9**]. Nevertheless, it seem to be more oriented to handle technological information than the usage situation and context reflected by this social data.

On the other side, there is in the recommendation a generic Data Component [**Figure 1.3.9**] that could support this information, but it is defined not only application-dependent but also as a component to handle the application data, and as we see in this topic, the support of contextual information (like the system's, social or emotional states or behaviors) could be considered as an emerging common need.

For this reason, it can be interesting to extend the MMI Framework & Architecture with a context management mechanism and to try to define some interfaces needed by the components currently recommended.

1.3.4.2 Emerging trends transversal to the criteria

On the other hand, we can identify some emerging trends that are transversal to the characteristics listed above.

The first transversal key topic (and unresolved from a standardization point of view) is the definition of models: twelve of our architectures propose interesting approaches concerning the modeling of the entities that participate in the multimodal interaction.

However, only SMARTKOM addresses the modeling task with a proposal coming from web semantic technologies, that can allow to evolve from a thesaurus, a list of properties or a structure, to a description of the dynamic relationships between entities.

68 See the proposal of a Session Management Component above.

69 For example, in Software as Services (SaaS) implementations of multimodal systems or in browser implementations.

In addition, depending on the modeled entity, the models are more or less expressive or homogeneous, and consequently, usable.

The modeling of the multimodal interaction phenomenon (FAME, CARE, ICARE, CICERO, SMARTKOM, HEPHAISTK, MEDITOR), the task (GALATEA, OPENINTERFACE, SQUIDY, ELOQUENCE), the dialog (REA, GALATEA, SMARTKOM), and the devices (FAME, ICARE, CICERO, SMARTKOM) is more extensive, tested and advanced than the modeling of the user (FAME, ELOQUENCE, REA), the application (OAA, SMARTKOM, ELOQUENCE, GPAC, HEPHAISTK) or the environment & context (SMARTKOM, CICERO) conceived to support and enrich the multimodal interaction.

This growing and common interest on models -expressed in SMARTKOM as a foundational principle[70], opens the way to reinforce the MMI recommendation with an effort to address this issue and to see how the MMI Framework & Architecture can respond to the data modeling needs.

The second transversal topic is distribution. It is tackled with solutions like the remote installation of triggers (OAA), the distribution of the *fusion / fission* mechanisms into nodes and components (OPENINTERFACE, SQUIDY) that can even be external to the *multimodal system,* the management of inputs as «sensed» data (input sensors) or as broadcasted media containing behavior (and interaction) information in the distributed streams (GPAC); and finally, the distribution of application services (SMARTKOM, HEPHAISTK).

This topic is also reflected on the service-oriented proposals of application services and services advertisement (OAA, SMARTKOM) and the networking services layer to manage the broadcasted input and output data of a rich application (GPAC).

The MMI Framework & Architecture reflects this topic in its distributed nature based on web standards. Nevertheless, there is no current implementations using the web services or a service-oriented approach from a distributed perspective.

The current stand-alone implementations are oriented to prototype mobile interfaces (Orange Labs), provide a multimodal mobile browser (Openstream), to test an authoring tool (Deutsche Telekom R&D) and to complete JVoiceXML, an open source platform for voice interpretation (TU Darmstadt).

We believe that it is possible that interesting extensions arise from a fully SOA implementation of the MMI Framework & Architecture standard according with its distributed nature.

A final emerging topic is the delegation of the interaction management by a client application. It is present in the form of application agents (CICERO,OAA) or application services [**Figure 1.3.25**] (SMARTKOM, HEPHAISTK).

The MMI Framework & Architecture does not deal with this subject because the application is meant to be the concrete implementation of the architecture.

A delegation approach supposes that an external functional core can delegate the management of the interaction to a *multimodal system* built in accordance with the standard, and providing multimodal functionalities to the client application.

This approach is not currently addressed, even if it could be the type of requirements of a multimodal browser, of a multimodal browser plug-in or a cloud application. It can be interesting to continue to explore the possible extensions that such approach could bring and how the MMI Framework & Architecture standard can support this type of implementation.

70 «...there should not be processing and presentation without an explicit representation»

1.3.4.3 Summary

This section compiled the results from our empirical study of the current *multimodal systems* under the light of a set of criteria that allow us to detect some emergent topics in the architectural design of this kind of systems. Then we compared these topics with the proposal of a standard multimodal architecture provided by the W3C, in order to find out some possible enhancements:

- In event propagation and handling
- In state management
- In decision support
- In context management
- In modeling from a multimodal perspective
- In service-oriented interfaces
- In user interface delegation

This empirical and qualitative analysis of the state of the art introduces the positioning of the Soa2m project, which will be described in the second part of this thesis.

Soa2m is the effort to address this emergent topics in the multimodal research, framed by the work on web open standards and more precisely the participation and contribution to the activity of the W3C's Multimodal Working Group. In the next chapter we will present the project, describe the proposal and explain the current implementations.

11 The Soa2m Project

The Service-Oriented Architectures for Multimedia (Soa2m) project is part of the Ubimedia joint research program between Alcatel-Lucent and the Institut Telecom (2009-2012) with the aim to explore future applications for the sector of digital media.

The focus was in the merging of communication means, data media and knowledge objects: the future will be one of the «Ubimedia», a digital environment in which the individual sends, receives and passes on information with the need of a extremely high level of continuity between ubiquitous communication – exchange of data anywhere, any time – and the digital media used for these exchanges, in order to create a dense, cognitive and always interconnected information universe.

The Soa2m project was one of the four research actions and it was build to propose strategies of semantic and smart interaction for the digital world described above.

In this context of research, the Soa2m research action intend to explore the Multimodal Instantiation of Assistance Services with the proposal of several supports for services in enterprise spaces using multimedia resources and input and output features in multiple *modes*.

Imagine a conference room where a series of seminars will take place. People enter and leave the conference room before, after and during the lectures. Before the meeting, the room access manager, sensing no human presence, is running in its waiting mode. This activity is interrupted when the door is "touched" by a badge. The user identity manager confirms the access rights for this person (the chair of the meeting) and for this type of event (structured conference). The access manager opens the door.

The system activates any available displays in the room and the event default services: the outline service, the notification service, the guiding service. The chair of the meeting is notified by a dynamically composed graphic animation, audio notification or mobile phone notification, about the services availability with a shortcut to the default outline service instance: a room FAQ.

The chair of the meeting selects the agenda instance of the FAQ service, and submits the agenda content with the presentation slides. The outline service composes an animated conference program and synchronizes its evolution state with the slides information. The outline service publishes this conference program (by default in the HDTV display and also available by internet or in nearest display devices), the conference alert service (for phone or pc text, image or audio input) and the washroom and coffee room location guide service as new services in the conference room

The chair of the meeting wants to setup the video projector. He selects the "How to set the video projection system?" question in the room FAQ on his mobile display and the assistance system proposes a how-to instance of the guiding service as a text with alternative dynamically discovered options. These options could be, for example: photo step-by-step instructions (iPhone, HDTV widget display, Website) or audio Instructions (Mp3 audio guide, Room speakers reproduction, HDTV audio).

The chair of the meeting chooses the room speakers reproduction, the guiding service is activated and he starts to set the video projector. Then some attendees arrive. The chair of the meeting changes to the slide show option and continues to follow the instructions at the same step it was paused but with another modality in another device.

The conference starts 30 minutes late and some attendees arrive at the room 1 hour later. They find the conference agenda service in any device with a highlight in who is being spoken and what topics have been covered till then. This will allow them to follow the conference without disrupting anyone. As they request any service after the start of the conference, the service response is always preceded by a notification service to remember attendees to put the mobile phone on silent.

During the break, attendees can consult the room FAQ and select the washroom guiding service while others can select the coffee room guiding service. The coffee room guiding service is activated by the outline service only in break time. When the conference resumes all the services instances linked to the outline service receive a notification: the attendees return to the conference room. Finally, the meeting draws to a close. As the chairman leaves the room, the access manager returns to its original state and the services instances are cleared.

Scenario 2.1: The Ubimedia Use Case

The Working Hypothesis of Soa2m

The working hypothesis was that it could be possible to build a system bringing services to each user with the use of the semantically best resources available, independently of the communication channel, the *mode* of restitution, the *media* or the device.

With this idea on mind, the more evident technological tool to adopt was the multimodality and then, the effort was oriented to find the best mechanisms to adapt the multimodal interaction to the context of use of this kind of services.

The Soa2m architecture proposal

As a first exploratory result, this thesis proposes an architecture of services enriched by semantic technologies and a set of tools for the dynamic discovery of multimodal features, based on open web standards and more precisely in the MMI Framework & Architecture recommendation of the W3C.

Our proposal begins with the case study presented in the last section. In complex situations, a case study uses one or more real-world examples in order to obtain a thorough understanding of the topic and if possible to draw lessons from the analysis.

Thus, the last section presented an in-depth examination of the design choices for multimodal architectures over a long period of time. This examination provided a systematic way of looking at these multimodal systems, analyze the architectural information against our needs, and report the conclusions.

As a result we gained understanding of how the multimodal systems architectures evolved; what might become important to look at more extensively; and what might be added to a hypothetical architecture of reference resulting from open standards.

Our case study was based on an information-oriented sampling technique and the sixteen critical cases were selected strategically in order to allow our final generalization[71]. This generalization led us to build some architectural hypotheses that compose the core of our approach: the proposal of a set of extensions to the MMI Framework & Architectures.

The Soa2m objective

This seeks to provide with the W3C's standard, the architectural support of emerging implementation requirements like were ours: the intelligent, dynamic and plastic adaptation of the user interface and the interaction cycle with the use of the best resources available to communicate a message and to generate a user experience in multiple *modes*, *modalities* and *media*.

Outline

In the following sections first, we will present the Soa2m architecture and its requirements, second, we will describe the semantic support envisioned in the architecture and third, we will describe the tools implementation. Finally we will conclude with the explanation of the relation between this proposal and the seven emergent topics outlined in the summary of the section 1.3.

2.1 The Soa2m Architecture.

A common view of a system's architecture is its structural definition [Bass et al. 1998]:

> *«the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them».*

This structural view is completed by a definition that includes the temporal evolution of the architecture, for example, according to a IEEE Standard [IEEE-1471 2000], a system architecture is defined as:

> «... *the fundamental organization of a system embodied in its components, their relationships to each other and to the environment and the principles guiding its design and evolution».*

71 See Supra section 1.3.4. Conclusion. The selection was made based on the representativity of the research domains, the focus on multimodal architecture issues and the completeness of architectural details in the published papers.

Finally to the temporal aspects of the design of a system architecture, is added the abstraction aspect. In this perspective [Shaw 1990] a system architecture:

«...is an abstraction of the run-time elements during some phase of the system's operation. A system may be composed of many levels of abstraction and many phases of operation, each with its own architecture».

This approach points out the encapsulation phenomenon, hiding some details of the system in order to better identify and sustain its properties. In this way, a complex system contains multiple levels of abstraction.

Perspectives on the architecture design.

Thus, generally the design decisions for system architectures cover various perspectives: the decomposition of the system's functionalities for the domain of interest, the determination of the structure of the system in terms of components and their interaction, and the allocation of functionality to each component.

The architecture of a large system can also be guided by using architectural styles and design patterns [Gamma et al. 1994].

According to [Garlan et al. 1994], an architectural style defines a family of systems in terms of a pattern of structural organization which includes:

- the components and connectors that can be used in instances of that style,
- the set of constraints on how they are be combined. For example, one might constrain the topology of the descriptions (e.g., no cycles) and execution (e.g., processes execute in parallel)
- an informal description of the benefits and drawbacks of using that style.

Architecture Styles

Architectural styles encompass the most important decisions about the architectural elements and emphasize the more important constraints on their elements and their relationships. As we have already see, a multimodal architectural style can be Pipes and Filters, Object-Oriented, Event-based, Repositories (blackboard), Interpreters or heterogeneous architectures.

On the other hand, design patterns like factories, observers, command and proxies are the common implementation methods that have been found repeatedly in software design.

Depending on the domain, the selection of an architecture style must respond to a series of requirements. For the Soa2m architecture, these requirements are divided in two categories: functional and non-functional.

As we see in section 1.2, the definition of a *multimodal system* can lead to a series of functional requirements (identified by the code **RF**):

RF1. The *mode*, *modality* and *media* integration (*fusion*) and restitution (*fission*),[72]

RF2. The bidirectional nature and the symmetry of the system,[73]

RF3. An appropriate real-time sensing and responding at the sensorial, functional and semantic levels,[74]

RF4. The management of the interaction cycles locally and globally,[75]

RF5. The support of incremental processing of incomplete and dynamic data,[76]

RF6. The integration of embodied and situated knowledge about the interaction situation; covering devices, users, domain and environment and context of usage.[77]

72 See supra section 1.2 Definition of a Multimodal System
73 See supra section 1.2.1.3 Interaction Management Strategy
74 See supra section 1.2.2.2 User Modeling
75 See supra section 1.2.1.2 The Multimodal Event
76 See supra section 1.2.1.3 Interaction Management Strategy
77 See supra section 1.2.1.4 Decision Making

These functional requirements are joined with 3 sets of the most common non-functional requirements for a software architecture. [IEEE-Glossary 1990] [IEEE-1061 1998]

Our proposal is around the addition of some enhancements to an existent standard and the provision of some tools to allow these enhancements. In this context, non-functional requirements become a very important part of our research strategy.

Evaluation strategy

The reason is that the MMI Framework & Architecture recommendation is presented as an architecture model for the multimodal domain, and as a model, it must be evaluated not only in terms of its functional requirements (that as we see can evolve over time) but mostly in terms of its non-functional requirements at its capability to be adopted by a large community of developers.

Outline

Thus, we will present in detail the non-functional requirements that will be the structure of analysis for the Soa2m proposal, because our proposal was motivated by these requirements and it will be confronted to them in the following sections. Thus, these requirements are our principal frame of reference and they must be defined to lay the foundations of our architectural choices and our contribution.

The first set, groups the requirements at design-time, identified by the code **RD**:

RD1. Extensibility

Taking into consideration future growth. It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension.

RD2. Flexibility

The ease with which a system can be adapted to changes. It describes how well the Architecture caters to the dynamicity; how, for example, the Architecture lends itself adapting to changing functionality, data models, users, *mode*, *modality*, *media* or even development methodologies.

RD3. Portability

The ability of a system to be deployed from one system to another.

RD4. Reusability

The likelihood of a system or system component of be used again to add new functionalities to another system, with slight modifications or no modification.

RD5. Modifiability

The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed.

RD6. Maintainability

The ease with which a software system or component can be modified to correct faults, improve performance, or other attributes, or adapt to a changed environment. And the ease with which a hardware system or component can be retained in, or restored to, a state in which it can perform its required functions.

RD7. Completeness

A system is complete if it entirely describes and specifies the system that exactly fulfills all requirements and contains all necessary information that is needed to implement that system. Completeness is both an external and an internal goal.It is external with respect to system requirements and it is internal with respect to the architectural intent.

RD8. Consistency

Ensures that different elements do not contradict one another (e.g. component and connector names, service names, interfaces, component functionalities and behaviors of interacting components)

RD9. Testability

Is the degree to which a system supports testing. A lower degree of testability results in increased test effort.

RD10. Auditability

Auditability is the ability of a software to keep track of what happened, who did it, from where, how and when.

The second set covers the requirements a run-time, identified by the code **RR**:

RR1. Interoperability

Interoperability is the ability of diverse and separate systems to work together. Is the ability of two or more systems or components to exchange information and to use the information that has been exchanged.

RR2. Performance

The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage. Concerning this requirement, predictability, not speed, is the foremost goal in real-time-system design. (e.g. latency and throughput)

RR3. Reliability

The ability of a system or component to perform its required functions under stated conditions for a specified period of time. Reliability is often measured as probability of failure ant the frequency of failures. It covers the failover management and the disaster recovery (i.e manual vs automatic)

RR4. Availability

It concerns the dependability, namely, the amount of trust that can justifiably be placed on the service it delivers. In this context, the availability is the degree to which a system or component is operational and accessible when required for use. (e.g. uptime, downtime, maintenance schedules)

RR5. Scalability

Scalability is the ability of a system to handle a changing amount of work in a capable manner or its ability to be enlarged to accommodate that growth. The scalability is made in three directions (out/up/down) and it can be functional (input/output load, system throughput, functionalities, amount of data, traffic volumes) or non functional (spatial, users, availability).

RR6. Security

The ability to be trustful, to keep secrets and to safeguard privacy. Also the ability to treat malicious attacks like a kind of faults, to include the ability to maintain some level of service in the presence of attacks, measured in terms of the success of global goal rather than in the survival of any specific system or component. (i.e authentication, authorization, data confidentiality, privacy)

RR7. Manageability

How efficiently and easily a software system can be monitored and maintained to keep the system performing, secure, and running smoothly.

RR8. Supportability

How effectively a software system or component can be kept running after deployment, based on resources that include quality documentation and diagnostic information.

The third set covers the requirements at the interaction-time, with the code **RI**:

RI1. Accessibility

Accessibility is the degree to which a system is available to as many people as possible whatever their hardware, software, language, culture, location, or physical: a diverse range of hearing, movement, sight, and cognitive ability. Thus, accessibility supports social inclusion for people with disabilities as well as others, such as older people, people in rural areas, and people in developing countries. (e.g.modality migration, access to partial data structures, internationalization, localization)

RI2. Usability

The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. Usability is a measure of how well users can take advantage of some system functionality.

In includes, for example, the learnability - easy to learn (e.g. novices can readily start getting some work done), the efficiency - efficient to use (e.g. experts have a high level of productivity), the memorability - easy to remember (e.g. casual users do not have to learn everything every time), the error rate -low error rate (e.g. users make few errors and can easily recover from them), the satisfaction - pleasant to use (e.g. discretionary/optional users are satisfied and like it); tradeoffs - depending on the situation, usability might be increased or decreased on purpose and the support of categories of users - depending on user experience, usability might have to be tailored to the user.

RI3. Configurability

Configurability is the capacity of a system to be efficiently extended, changed, parametrized and refined for use in a particular context. It ensures the ease of install, and the control over appearance and behavior in a given system. It can be expressed in terms of parameters, features and profiles. A feature is an aspect of a system meaningful in a specific context. Features allow to decompose a configuration into usable chunks for a given context that can be expressed by a profile.

RI4. Buildability

It defines the ease and efficiency with which structures can be built, and the extent to which the design of a system facilitates ease of construction.

RI5. Automation

The capacity of a system to increase productivity and/or quality beyond that is possible with current human labor. It optimizes productivity in the production of goods or in the delivery of services.

RI6. Integrability

Bringing a system into another system. Linking together different systems and software applications physically or functionally, to act as a integrated whole. System integration is also about adding value to the system, capabilities that are possible because of interactions between subsystems.

Under the light of these requirements, in the following section we will describe the Soa2m architectural style, inherited from its adoption of the MMI Framework & Architecture as a starting point.

This architecture is also build according some commonly accepted design patterns that we will describe in detail in section 2.3. The description will follow the 4+1 Architecture view model presented in [Krutchen 1995].

2.1.1 Soa2m Logic View

The seven generalized results from the case study of recent multimodal architectures[78] drives us to some architectural conclusions.

The first four resulting topics to treat, namely, event propagation and handling, state management, decision support and context management; implies the proposal of dedicated components or the redefinition of some existing components in the MMI Framework & Architecture recommendations in order to achieve some completeness **(RD7)** in regard to the current types of multimodal implementations.

The Soa2m architecture extensions

In order to support this changes, the Soa2m proposal begins by redefining the basic elements of the architecture to ensure consistency **(RD8)** and then, it adds a new layer, the service layer, which is the Soa2m start point to resolve the three remaining requirements.

First, it addresses the exploration of service-oriented interfaces. Second, it builds the foundations for the user interface delegation, to achieve integrability (**RI6**). And finally, it provides a data access mechanism to the modeling process from a multimodal perspective. This collaborates to achieve : flexibility (**RD2**), completeness (**RD7**), scalability (**RR5**), accessibility (**RI1**), configurability (**RI3**) and buildability (**RI4**).

Therefore, in the Soa2m architecture some components are added to the original MMI Framework & Architecture structure and a service abstraction is proposed.

2.1.1.1 Basic Elements of the Architecture

In Soa2m the modules represent the more basic abstract element[79] of the architecture, with a separation of concerns, enforcing logical boundaries between processes and containing the inheritance structures. Nevertheless, each module is a black-box, for which we only know the function, and not the implementation.

The Soa2m MMModule

A basic Soa2m element is called the **MMModule**. [**Figure 2.1.1**] All the MMModules in Soa2m are composed of a communication layer and a data layer, and they use the same event protocol.

Each layer corresponds to a set of common APIs. In one hand, to allow the communications of the module with external entities (e.g. services) and in the other hand, to allow the access to modeling (e.g. metadata) and application data, and the storage of some of these data.[80] In this way, we begin to respond to the requirement of modeling and service orientation coming from our emergent topics.[81]

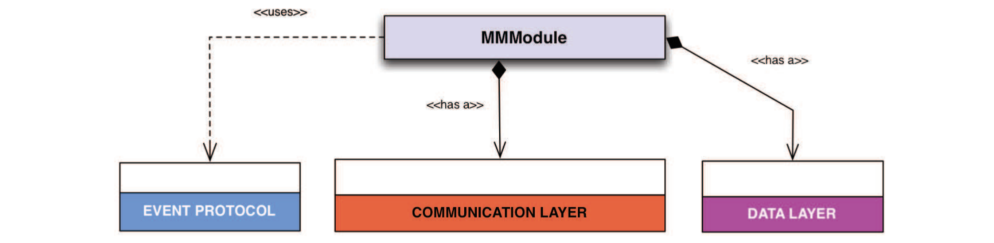


Figure 2.1.1: The MMModule and its mandatory layers in Soa2m.

The Soa2m MMModule

From this abstract and very basic element, the Soa2m architecture derives two concrete categories of modules: the **Component** and the **Manager.** [**Figure 2.1.2**]

78 See supra 1.3.4.3 Summary
79 An abstract element is an entity which does not exist at any particular time or place, but rather exists as a type of thing (as an idea, or abstraction). It is a is a class which cannot be instantiated, or in other words, a class that does not have any concrete existence. The mechanism of abstract classes allow to define behaviors (methods) whose implementation must be done by the inheriting classes.
80 We will describe later these APIs with more details in section 2.1.3 after we briefly discuss this logic view.
81 See supra 1.3.4.3 Summary

A **Component** is a module that encapsulates a set of related functions, can communicate with other components and in some cases, with modules external to the *multimodal system*. A component provides functionalities, exported and used through its interfaces.

Each **Component** can contain a number of separate processes encapsulated in **Managers**, and works independently to another **Component**. A **Component** have few interaction with a **Manager** except in the sense that one Component may use a **Manager**, to achieve its purpose. [Figure 2.1.2]

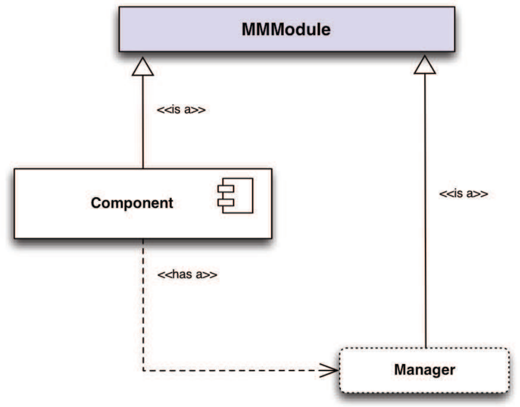


Figure 2.1.2: Basic categories of Modules in Soa2m.

In addition to the basic modules, Soa2m proposes a service-oriented perspective.

The Soa2m service

For us, a service is a well-defined advertised functionality that is built on the top of any kind of module. Then, modules (components, managers) are more finely grained than services.

A service implies that a consumer selects a supplier to fulfill its needs and the supplier accepts to provide the service with some conditions reflected in a service contract. Consequently, the supplier must provide a way for consumers, to be aware of the available services. This is called the service advertisement.

Justification of the use of Soa2m services

For us, a service is a well-defined advertised functionality that is built on the top of any kind of module. Then, modules (components, managers) are more finely grained than services.

Given our initial need to use the best resources available to communicate, such advertisement mechanism is a basic requirement in the Soa2m architecture.

This justifies the inclusion of a service structure to our proposal: to select dynamically the available resources and evaluate them, some mechanisms of advertising and discovery descriptive information are needed.

These mechanisms are already defined in service oriented architectures and in web services technologies.

The main features of service-oriented systems are their flexibility in handling dynamicity and their suitability for the integration of new modules. Services can appear or vanish on a network, notifying the whole system. Service-oriented systems allow robustness, coordinating services in a programmatic decentralized collaboration.

The CORBA [OMG-CORBA 2001] standard, was the precursor of services-oriented architectures, enabling different languages and different computer architectures to share data and act in the same application. Later, Web services came up, providing this kind of wider software interoperability using Web standards.

So, the service is the core element in services-oriented architectures.

It is defined as «*a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description*» [OASIS-SOA 2006]

The client (consumer) of a service (supplier), has no need to know where a service is executed – a property named *location transparency.*

Furthermore, the service's clients doesn't need to know what is the service execution platform – services are *technology neutral* – as long as they can interact with the service in a standard way. It is an effective mechanism for data and information integration on distributed systems like the web.

In addition, as we already say, a service provides a contract defined by a description and by one or various interfaces. This allows the change of the service implementation without reconstructing the client as long as the contract is not changed.

Soa2m services can be composed

A service can be used as stand-alone mode or it may be integrated in a higher-level service (by composition) distribution. This promotes reusability (**RD4**). Legacy applications can be transformed in services by using some wrapper techniques.

Services communicate with their clients by exchanging messages. Typically, the request/response message pattern is used.

From the client point of view, a synchronous or asynchronous communication mechanism can be implemented, it is not fixed a specific communication protocol.

They communicate with other services and clients using standard and decoupled message-based methods such as XML messages exchanges. This characteristic is called loose-coupling.

Services can participate in a workflow (called service choreography), which is the movement of information or tasks through a work process. With this goal, services need to be discovered by clients. This mechanism is provided by a service directory (or service registry) in which a supplier can publish (register) or advertise his service.

As a result, a services-oriented architecture posses three building blocks:

- The service provider that created the service and may publish their interfaces and access information to some registry.
- The service registry, also known as service broker; responsible for making available the interfaces and access information of the implementation to any potential service consumer.
- The service consumer or client binds to the service provider in order to invoke the service. It can be done by locating entries in some registry using various search operations or by knowing the service's endpoint by any other means.

The Soa2m architecture is designed to support a service provider implementation in order to achieve integrability (**RI6**).

Outgoing and Shy services in Soa2m

A service can be of two types: an **Outgoing Service** or a **Shy Service.** 12 [**Figure 2.1.3**]

An **Outgoing Service** is a service that advertises its operations and that can be requested and communicate with external entities or other **Outgoing Services** in the *multimodal system*. Therefore, an **Outgoing Service** can be registered in a service registry and provide a detailed description of its behaviors and attributes.

In contrast, a **Shy Service** is a service that does not advertise its operations. Its endpoint is known as its behavior and interfaces. It only communicates internally to the system.

The definition of this kind of service reduce the working load coming from service description and advertisement in the implementation, especially for local subsystems.

Hence, this design decision is made in order to ensure a better buildability (**RI4**) keeping the integrability of the subsystem (**RI6**) and the extensibility of the *multimodal system* (**RD1**) at the same time.

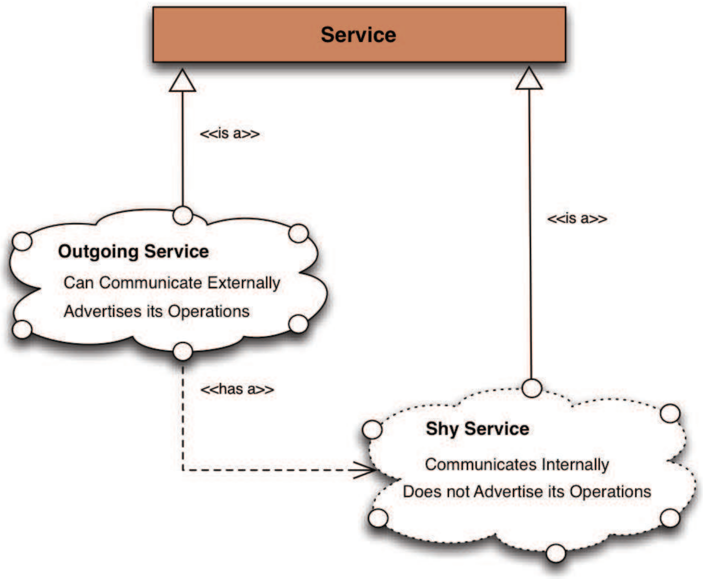

Service
<<is a>>
<<is a>>
Outgoing Service
Can Communicate Externally
Advertises its Operations
<<has a>>
Shy Service
Communicates Internally
Does not Advertise its Operations

Figure 2.1.3: Basic categories of Services in Soa2m.

Generally, in Soa2m, an **Outgoing Service** will wrap a **Component** to add the advertising and the distributed communication mechanisms to it, while a **Shy Service** will wrap a **Manager** to add only the communication mechanism.

This aspect is defined in order to keep the consistency (**RD8**) in the system and to contribute into this service wrapping proposal that provides interoperability (**RR1**), extensibility (**RD1**), flexibility (**RD2**) and modifiability (**RD5**).

Conclusion

To sum up, in Soa2m the more basic element is an abstract entity called **MMModule**. This entity contains a series of layers that will be predefined API's for all the types of modules.

Two kind of modules are defined, the **Component** and the **Manager**, which is a helper sub-system to achieve the **Component** tasks.

These modules are wrapped by entities named Services, which are used to add a communication and advertisement features.

The **Outgoing Service** can advertise its behavior and interfaces and communicate externally while the **Shy Service** does not advertise but shares the same communication mechanism. [**Figure 2.1.3**]

It is important to point out, that not all components are wrapped by services because some components are only clients and not service providers, and because not all implementations can support services technologies.

It depends on the implementation and deployment of the multimodal component. This is important to respect the portability (**RD3**) of the *multimodal system*. [**Figure 2.1.4**]

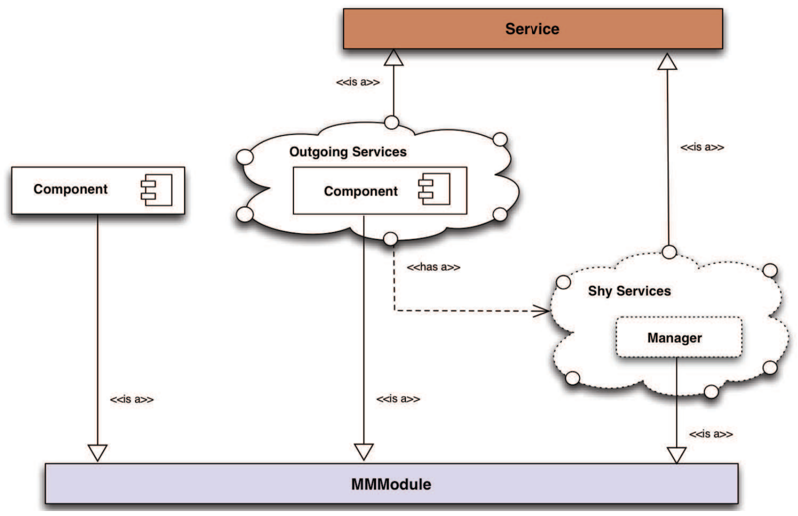


Figure 2.1.4: Modules are wrapped by services in Soa2m.

2.1.1.2 Abstract Layers of the Architecture

In the Soa2m architectural logic view, the abstraction layer is the representation by which data and operations are defined with a representation similar in form to its meaning, while hiding away the details of the implementation.

The low-level layers expose details of the computer hardware where the system is running, while high-level layers deal with the operational and business logic of the system.[82]

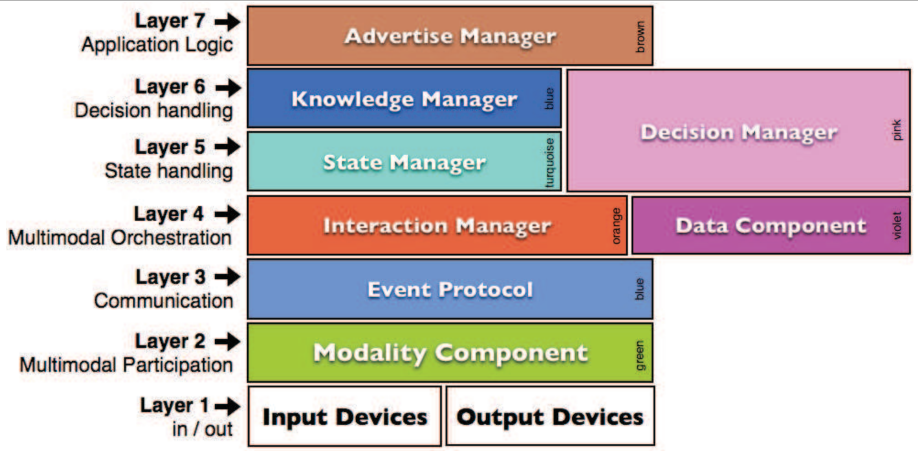


Figure 2.1.5: Abstract Logic Layers in Soa2m.

- In the **layer 1: in/ out** [**Figure** 2.1.5] we find the input/output devices in charge of the input signal capture and the output generation; and the low level event handling (e.g. low-level events can represent window-system occurrences or window input/ output).

- The next layer is the **layer 2: Multimodal Participation**. [**Figure** 2.1.5] This layer represents the Modality Components and is adopted from the MMI Framework & Architecture.

In a *multimodal system*, the data received from a particular device may be processed at multiple levels of abstraction. The Modality Component is the first of them in the Soa2m architecture.

82 This description is inspired on the Proposal of a Hierarchical Architecture for Multimodal Interactive Systems made by [Araki et al. 2007]

The Modality Component definition

The Modality Component abstraction in the Soa2m project responds to the need of context-awareness. It depends on the user and the context of use whether a device is suitable for an intended interaction (and its affordance). Thus, the design and implementation of the interaction cycle should not be restricted to specific hardware or input and output: the design of the multimodal interaction requires an abstraction layer for the user interface in which inputs and outputs can be switched or overlapped.

For example, if the physical shape of an equipment causes errors in operations, the user must be allowed to switch to other *modality* (e.g. to another device) more aligned with his personal attributes and capabilities. This is also the case for some type of tasks or environments, that could need a more suitable modality as the context changes.

This implies crossing interaction styles and designing the interaction with a dependance on the meaning rather than on a concrete device.

This is the goal of the Modality Component abstraction layer in the Soa2m architecture, with the intention to respond also to the functional requirement concerning the support of *modes*, *media* and *modalities* (**RF1**) and the functional requirement of integration of the environment and context information (**RF6**).

It also enhances the flexibility (**RD2**), scalability(**RR5**), consistency (**RD8**), portability (**RD3**), accessibility(**RI1**), usability(**R12**), buildability (**RI14**) and interoperability (**RR1**).

Modality Components are virtual or logical devices, physically separated from the controlled material service and from the target application. They are defined in terms of the meaning or in terms of interaction participation and not in terms of the state of the application or the hardware (e.g. a commander, a pointer or a listener).

In addition,the rapid rate of innovation in new modalities precludes a piecemeal, modality by modality, standardization process. An extensible approach (**RD1**) like the MMI Framework & Architecture and its a generic model of modalities (and adopted by Soa2m) will prevent for example, the explosion of modality descriptions of multimodal capabilities like, in the case of the W3C, HTML5 Speech, HTML5 handwriting, HTML5 biometrics, HTML5 stereoscopics, HTML5 SpatialSound and so on, with no foreseeable end in sight. The Modality Component unifies the description of the common attributes in all these cases to provide a single discovery tool to be followed and completed by a more detailed and specific description.

As virtual devices, the Modality Components are the source of meaningful input and output without constraints for physical shape of the device, type of user interface or interaction style.

Abstraction levels in Modality Components

A Modality Component can also encompass multiple levels of abstraction in its internal data processing as we explained in sections 1.1.6 Fusion and 1.1.7 Fission. On the input side, for example, speech input can be handled as signal data, or described as a series of utterances, or interpreted as a sentence with some meaning. On the output side, a vocal message may be synthesized from an abstract representation of meaning by a natural language engine, from a text or from recorded sound.

In both cases each representation corresponds to a particular level of abstraction and according to the implementation of the Soa2m Modality Component, it can provide one or various of these data abstractions.

Thus, this layer provides an abstract API for *modes*, *modalities* and *media* and required in (**RF1**).

In the MMI Framework & Architectures (and in Soa2m), a single input/output device can be involved in several Modality Components, and a Modality Component can be deployed as an abstraction over multiple input/output devices. The Modality component also represent a black-box that can be made of several other components.

In contrast, in Soa2m the input of this layer is modality-dependent and the output is a service interface (providing results) or a service request (communicating results) depending on the implementation. This aspect allows the modifiability (**RD5**) and the reusability (**RD4**) of these modalities, by the proposal of a common mechanism of interface and request, which is the persisting front-end of a component that can change.

- The **layer 3: Communication** [**Figure 2.1.5**] is inherited from the MMI Framework & Architectures, and corresponds to the protocol of communication including the Interaction Life-Cycle Events.

The MMI Interaction Life-Cycle Events are used to communicate in Soa2m

To send the Interaction Life-Cycle Events, service requests are used. To receive the Interaction Life-Cycle Events the interface of the service wrapper of each **Component** or **Manager** is used. These events are asynchronous and all the modules in Soa2m, namely **Components** and **Managers**, are able to handle events that are delivered to them asynchronously. This ensures the consistency (**RD8**) requirement because all the components have an unified way to communicate.

- All the events that Modality Components generate are dispatched to the **layer 4: Multimodal Orchestration**, containing the Interaction Manager that collaborates with the Data Component [**Figure 2.1.5**]

In the Data Component we locate the storing component for the interaction cycle. These are components to access the data corresponding to the model in the MVC-pattern. It stores for example, the transactional data and the knowledge data: the participants, situation and environment models.

The Interaction Manager in the Soa2m architecture has two main roles: it is responsible of the interaction cycle and responsible of the integration and composition of *modalities* an *media* (the functional requirement **RF1**).

The responsibilities of the Interaction Manager in Soa2m

First, the Interaction Manager in Soa2m can be viewed as a temporal compositor, a dialog manager, a scheduler, a planning engine, a task manager, in short, a functional block responsible of the temporal synchronization of modalities at a semantic level. It handles the evolution of the interaction cycle and the turn taking mechanism (e.g. with state transitions).

Second, the Interaction Manager in Soa2m is also a meaning generator. A meaning is a significant information that is created by a system submitted to a constraint when it process an information that has a connection with the constraint. The meaning is formed of the connection existing between the information and the constraint of the system. The function of the meaning is to participate to the determination of an action that will be implemented in order to satisfy the constraint of the system.

The Interaction Manager generates meaningful input commands (or notifications) on one side, by integrating the information received from the Modality Components according to constraints previously defined or provided by a decision mechanism.

It also generates output multimodal performances that are meaningful. A performance is an act of staging or presenting a form of expression, generally combining multiple media.[83]

The Interaction Manager is a meaning generator

In other words, as a meaning generator, the Interaction Manager in the Soa2m architecture is responsible of the *fusion*[14] and the *fission*[15] mechanisms, which respond to the functional requirement expressing the need of bidirection and symmetry of the system (**RF2**) and the need of appropriates responses at the semantic level (**RF3**).

This facet of the Interaction Manager is then responsible of modality *fusion* for input processing and modality *fission* for output processing. Since there are various strategies of modality *fusion / fission* handling this layer do not restrict the strategy.[84]

The Interaction Manager receives multiple modality specific input from the lower layer and passes the interpreted command message to the upper layer. It receives an output message from the upper layer, and passes the modality specific output definition to each instance concerned in the lower layer.

The input of this layer is a service request and the output is a service interface (publishing commands) or a service request (communicating commands) depending on the implementation.

83 We prefer the term performance to the term presentation because a performance is more adapted to a situated experience: it is something that «happens» in a tridimensional reality with multiple participants and a spatial context (the stage). A performance can change on every new instance. A presentation is a more static view.

84 See supra section 1.2.1.3 Interaction Management Strategy

- In the **layer 5: State Handling**, participates the State Manager and the Decision Manager **[Figure 2.1.5]**.

Responsibilities of the State Manager

The State Manager in the Soa2m Architecture is responsible of the management of the evolution of the interaction cycle and the modifications in any of the participants of the system that could affect the interaction cycle. This component is also aware of the system's capabilities, like the available modalities or the processing state.

This responds to the functional requirement about the management of the interaction cycles locally and globally (**RF4**), the requirement of an appropriate real-time sensing (**RF3**); and partially, to the requirement concerning the support of processing of dynamic and incomplete data (**RF5**).

The State Manager delivers state information about the interaction cycle and its participants and it can store external and internal phenomena traces in three data structures [Bach 2007] depending on the complexity of the implementation:

- the image of the current state, represented by the current situation and its state metadata.
- the history of states, represented by a set of situation episodes and its state metadata.
- and the expected states, represented by the extrapolation of the current situation in the future according to a desired outcome and the stored episodes; and its state metadata.

To accomplish this activity, the State Manager collaborates with the Decision Manager, who is responsible to apply the decision making strategies[85] and publish or communicate its results (this depends on the requester).

For example, the Decision Manager collaborates to select the expected state based on : the desired outcome, the targeted time frame, the stored episodes and the knowledge about the context needed to make the decision. It returns a list to the State Manager that eventually select the best option in the list and stores it.

The decision Manager also collaborates on the construction of episodes, by supporting the selection of patterns of situations according to previous knowledge, the focus and the state metadata.

- As explained in the example above, the Decision Manager in the **layer 6: Decision Handling**, collaborates with the **layers 4**, **5** and **7**; and implements and manages the decision making strategies[86] **[Figure 2.1.5]**

The decision strategies can be very simple

These strategies can be as simple as a set of rules, or more complex, based on Artificial Intelligence Technologies. It will depend on the implementation.
To make decisions, the Decision Manager relies on the Knowledge Manager which is responsible of handling all the knowledge sources and generating knowledge.

This is a provider of understanding static and dynamic data to the Decision Manager by using models described with semantic technologies, taxonomies, thesaurus or only some metadata files, like key-value properties lists. It also can generate knowledge by collecting data, making inferences and learn depending on the implementation requirements.

85 See supra section 1.2.1.4 Decision Making
86 See supra section 1.2.1.4 Decision Making

The Knowledge Manager and Decision Manager proposals respond to the functional requirement about the integration of the situated and embodied knowledge (**RF6**). It also collaborates to a proposal focused on the completeness (**RD7**) that can be sought for a reference architecture like the MMI Framework & Architecture Recommendation. And finally it can enhance the accessibility (**RI1**) and configurability (**RI3**) of multimodal systems .

- In the **layer 7: Application Logic**, we propose the Advertise Manager [**Figure 2.1.5**]. This is the upper-most layer for the invocation and advertise[87] of the interaction by determining the interaction and performance strategies (the performance of modalities is the «instantiation» in the sense given by the ELOQUENCE architecture).

The Advertise Manager is a registry

On one hand, the Advertise Manager is a registry to give access to the services provided by the *multimodal system* to external entities (other systems or client applications) and internal entities (**Component** and **Manager**) if needed. With this goal the Advertise Manager provides the information necessary to choose the target services (the service description), to configure these services and to consume them providing a web API . Thus, the Advertise Manager is wrapped by an **Outgoing Service.**

There are many ways to consume the Advertise Manager's services. In the first case, the consumer and provider entities become known to each other (or at least one is known). This is the case in the communication between **Outgoing Services** and **Shy Services** in Soa2m.

In a second case, consumer and provider entities have somehow a previous agreement on the service description and semantics that will govern the interaction between them. This could be the case in the communication between **Outgoing Services** and **Client Applications** for the delegation of a multimodal interface management to a Soa2m architecture.

In the third case, the service description and semantics are exchanged dynamically by the consumer and provider to arrive to a service agreement before starting the interaction.

This could be the case in the communication between **Outgoing Services** and external entities when a multimodal application is build with the Soa2m architecture. Some of these steps may be automated, others may be performed manually.

The Advertise Manager is discovery service

Then, in the Soa2m architecture, the Advertise Manager is a discovery service, that can be used to publish and search for descriptions meeting certain functional or high-level (semantic) criteria. It is also used by **Component** entities to advertise their service descriptions.

In this aspect, this layer is a response to the functional requirement concerning the global management of the interaction (**RF4**) and also non-functional requirements like the scalability (**RR5**), the reliability (**RR3**), the availability (**RR4**), the interoperability (**RR1**) and integrability (**RI6**), the auditability (**RD9**) and the automation (**RI5**).

For dynamic discovery, the consumer may interact directly with the discovery service to find an appropriate provider to engage. For static discovery, a human may interact with the discovery service through a browser.

On the other hand, the Advertise Manager is also responsible of the generation of the description of services. It generates service descriptions annotated with semantic web technologies.

The Soa2m conceptual model

These abstract layers proposed by the Soa2m architecture, are based on the conceptual model of a *multimodal system* as a system build in analogy of some functional blocks detected on humans.

This design decision is founded on the premise of simulation; it supposes that for a more natural interaction with humans, the system must be build (in some cases, e.g. humanoids and robots) and must behave «**like**» if it was a human.

87 To advertise in Soa2m is to assert a thing with a purpose. Behind the advertisement there is an motivated intention that can be described in a contractual way. This is a commitment expressed by a proposal or a contract. It has also the meaning of divulgation, promote, declare. It also implies disclosure and trust.

The goal is the naturalness of the communication between the user and the system, based on technological achievement and the ability to emulate how humans interact with each other.

With this objective, some of the recent architectures, studied in section 1.3, were build following the three first levels of the conceptual model depicted in [**Figure 2.1.6**]: they addressed in their building blocks the **sensory**, **functional** and the **semantic** level[88]; the coverage of each level depending on the domain and needs of each project.

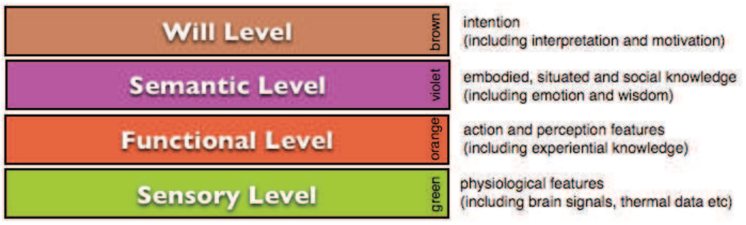

Will Level
brown
intention
(including interpretation and motivation)
Semantic Level
violet
embodied, situated and social knowledge
(including emotion and wisdom)
Functional Level
orange
action and perception features
(including experiential knowledge)
Sensory Level
green
physiological features
(including brain signals, thermal data etc)

Figure 2.1.6: Conceptual Model of the Soa2m Multimodal System.

The fourth level represents the intentionality and its background. [**Figure 2.1.6**] It appears as a dedicated component only in the proposals of SMARTKOM (Intention Analysis Module)[89], REA (Understandig Module)[90] and ELOQUENCE (Instantiation Module)[91].

A layer for the management of the intention

This is the intentional facet of the interaction, that implies the original motivation to execute a *communicative act.*[22] The motivation defines the final instantiation of the message and its interpretation (this is the received message filtered by the will of the receiver). [Sfez 1992]

As we know for humans, the will is a contextual notion. It concerns what I **want** to do, but also what I **can** do in the current situation and what I **must** do according with some authority. The will is usually advertised using words, but it can be also expressed by actions.

For this reason, in Soa2m we propose a layer covering the explicit and contractual expression of the intent (the commitment) of features and services. In Soa2m it includes the notion of «**full disclosure**» which means, the affirmation of what a functional block or participant does **not want** to do, or what it **can not** do.

These four levels are reflected in the previously described abstract layers of Soa2m and they are also applied in the representation of data, namely the user and situation models that will be presented later.

In this way, the Soa2m conceptual model addresses the functional requirement: **RF6** the integration of embodied and situated knowledge about the interaction situation; covering devices, users, domain and environment and context of usage. It also collaborates with the interoperability (**RR1**), the supportability (**RR8**), the reliability (**RR3**) and the availability (**RR4**).

2.1.1.3 Structure of the Soa2m Architecture

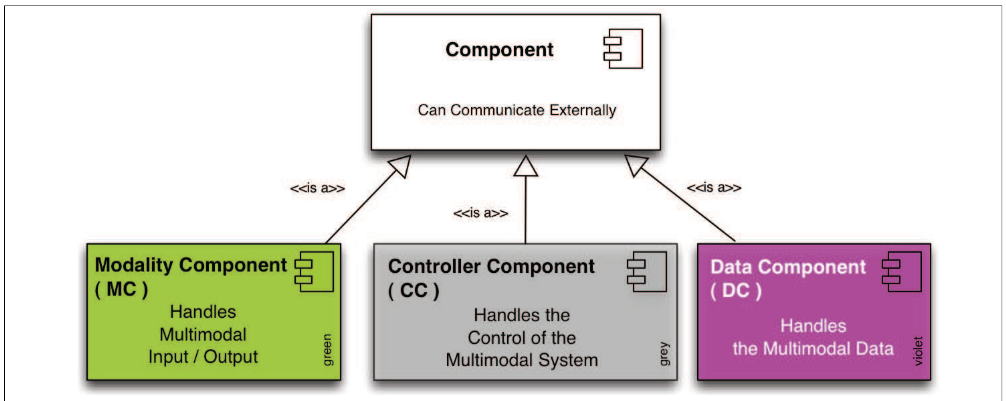
In this section we will present the structural relations and the building blocks proposed to handle the functionalities of these layers.

As we presented is section 2.1.1.1 Basic Elements of the architecture, the Soa2m architecture is composed of **Components**, **Managers** and **Services**. These basic elements are distributed according to the MVC-pattern and the MMI Architecture and Interfaces recommendation.

88 For an introduction of these levels see supra section 1.2.2.2 User Modeling.
89 See supra 1.3.3.4 SMARTKOM - 2003
90 See supra 1.3.3.2 REA - 1999
91 See supra 1.3.3.6 ELOQUENCE - 2004

First, the architecture comprises three categories of Components: the Modality Component (**layer 2**[22]), the Controller Component (**layer 4, layer 5 and layer 7**) and the Data Component (**layer 4 and layer 6**). [**Figure 2.1.7**]

Proposal: Categories of Components for the Soa2m architecture



Component
Can Communicate Externally
<<is a>>
<<is a>>
<<is a>>
Modality Component (MC)
Handles Multimodal Input / Output
green
Controller Component (CC)
Handles the Control of the Multimodal System
grey
Data Component (DC)
Handles the Multimodal Data
violet

Figure 2.1.7: Component Categories in Soa2m.

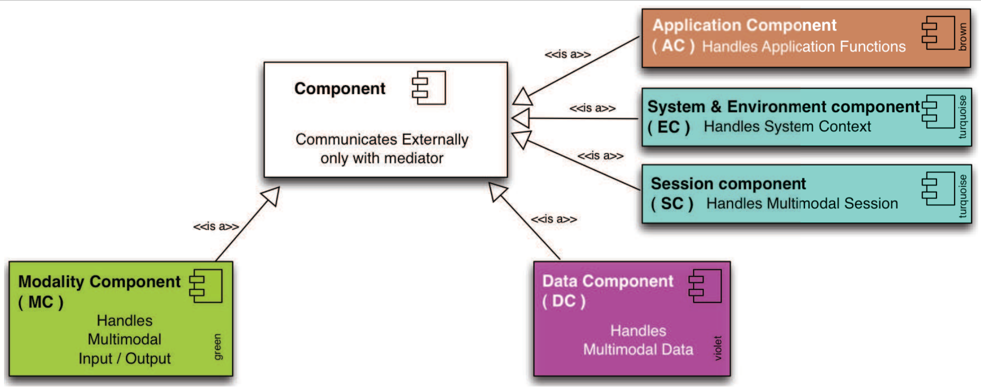
The Modality Component and the Data Component are inherited from the MMI Framework & Architecture recommendation. [**Figure 2.1.7**] The Controller Component is an extension to the standard. This component encompasses all the control blocks of the Architecture and integrates the Interaction Manager of the MMI Architecture and Interfaces proposal.

This choice is made because in a *multimodal system* the control is not limited to the interaction handling. State and instantiation handling are complementary parts of the control of interaction cycles. It was also made to respect the requirement of consistency (**RD8**), learnability (**RI2**) and buildability (**RI2**).

Proposal: a controller for the system and the interaction

Thus, in Soa2m, «control» means: the control of the inputs / outputs, their interpretation, their instantiation, their states, their availability, their discovery and the control of the interaction cycle. In consequence, the Controller Component fulfills all these functions.

From this perspective, the Soa2m architecture is a centralized system. Nevertheless, as we will see in section 2.1.1.4, given the Russian Dolls principle of the MMI Framework & Architecture recommendation, the centralization issues are mitigated by the distribution of responsibilities into imbricated components and the proposal of services.



Application Component
(AC) Handles Application Functions
brown
System & Environment component
(EC) Handles System Context
turquoise
Session component
(SC) Handles Multimodal Session
turquoise
Component
Communicates Externally only with mediator
<<is a>>
<<is a>>
<<is a>>
<<is a>>
<<is a>>
Modality Component (MC)
Handles Multimodal Input / Output
green
Data Component (DC)
Handles Multimodal Data
violet

Figure 2.1.8: Component Categories in the MMI Framework & Architecture.

Proposal: A set of Managers for the Soa2m architecture

Second, the Soa2m architecture proposes four Managers in addition to the original proposal of the MMI Framework & Architecture [**Figure 2.1.9**], which suggest an Interaction Manager participating on the **layer 4** [22] and responsible of the control on the turn-taking and the integration and composition of *modalities* and *media* (which is the first functional requirement **RF1**[23]).

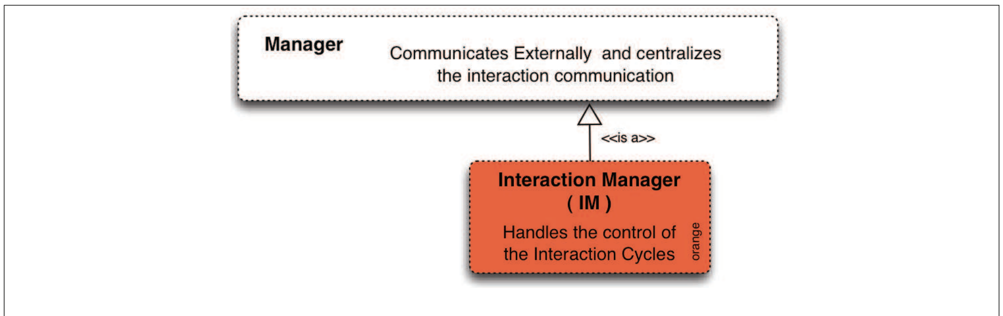

Manager
Communicates Externally and centralizes the interaction communication
<<is a>>
Interaction Manager (IM)
Handles the control of the Interaction Cycles
orange

Figure 2.1.9: Manager Categories in MMI.

These proposed four managers are [**Figure 2.1.10**] : the Advertise Manager which is responsible of **layer 7 Application Logic**, the State Manager that participates in **layer 5 State Handling**, the Knowledge Manager that participates in **layer 6 Decision Handling** and the Decision Manager which participates in **layer 5 State Handling** and **6 Decision Handling**.

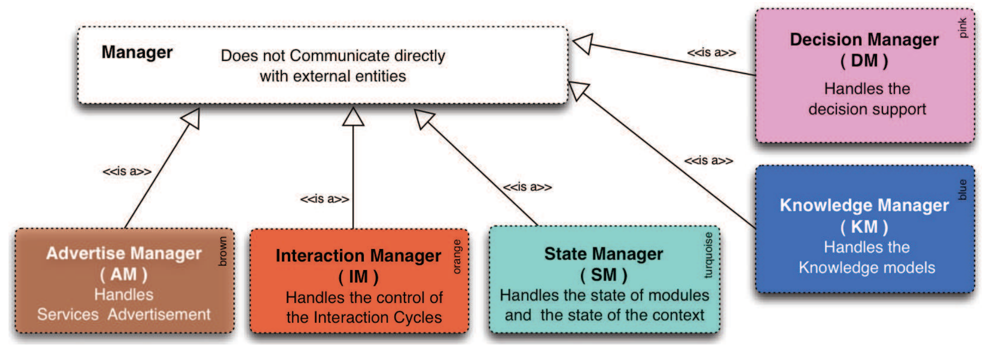

Manager
Does not Communicate directly with external entities
<<is a>>
<<is a>>
<<is a>>
<<is a>>
<<is a>>
Decision Manager (DM)
Handles the decision support
pink
Knowledge Manager (KM)
Handles the Knowledge models
blue
Advertise Manager (AM)
Handles Services Advertisement
brown
Interaction Manager (IM)
Handles the control of the Interaction Cycles
orange
State Manager (SM)
Handles the state of modules and the state of the context
turquoise

Figure 2.1.10: Manager Categories in Soa2m.

As Managers all these modules are subsystems. They represent a coherent unit with an interface with the hosting Component. They are also a partition of the hosting Component responsibilities, with defined boundaries.

In Soa2m these boundaries are functional, but also reflected in the implementation and structure of the source code in packages and libraries[92].

The principal reason to propose a set of managers who recovers some responsibilities awarded by the MMI Framework to three generic components, concerns the functional requirements of Soa2m.

These are: the management of the interaction cycles locally and globally (**RF4**) and the consistency (**RD8**), the buildability(**RI4**), the extensibility(**RD1**) and the scalability(**RR5**).

Firstly, we have a conceptual reason. The MMI Architecture and Interfaces is based on the MVC pattern[93] and support both distributed and co-hosted implementations. The architecture also proposes an imbrication mechanism :

92 See infra section 2.3 The Soa2m Tools Implementation
93 See supra section 1.3.1.1 The Interaction Model in the MMI Architecture

«*Since the internals of a Component are hidden, it is possible for an Interaction Manager and a set of Components to present themselves as a Component to a higher-level Interaction Manager. (...) The result is a 'Russian Doll' model in which Components may be nested inside other Components to an arbitrary depth. Nesting components in this manner is one way to produce a 'complex' Modality Component, namely one that handles multiple modalities simultaneously*.» [W3C-MMIA 2012]

In distributed scenarios, each of these nested components can need to react to global context and environment changes, and also they can need a support to handle the local state and context. The recommendation, in its current state, does not address this issues, leaving this question aside:

«*The Runtime Framework is a cover term for all the infrastructure services that are necessary for successful execution of a multimodal application. This includes starting the components, handling communication, and logging, etc. For the most part, this version of the specification leaves these functions to be defined in a platform-specific way*» [W3C-MMIA 2012]

According to this position, the context and state handling are considered «infrastructure services» related indirectly to the interaction cycle management and mostly concerning run-time system issues (previously defined in the MMI Runtime Framework proposal). [**Figure 2.1.11**]

Nevertheless, in an user-oriented, situated and embodied perspective, the context becomes a vital part of the interaction design and management[94], and as we show in section 1.2 and 1.3, this context can have multiple granularities and types of influence in the behavior of a *multimodal system*.

Secondly, leaving this important part unresolved for multimodal designers or platform producers, the ease and efficiency with which the multimodal architecture can be built, decreases, and the design recommendation does not facilitate ease of construction.

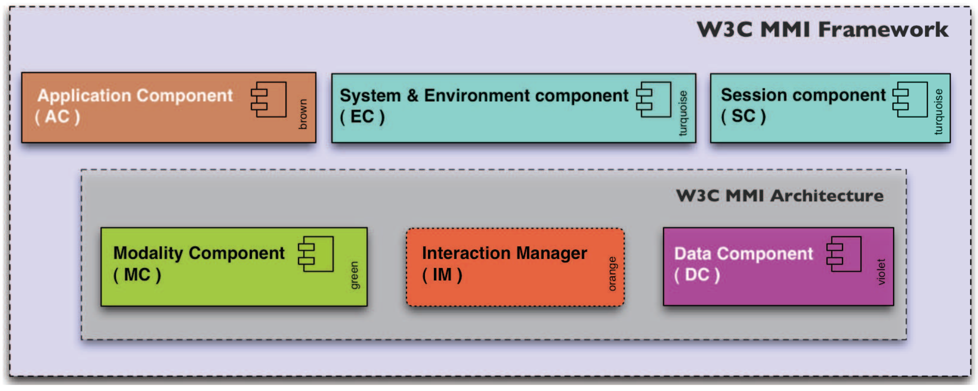


Figure 2.1.11: The MMI Modules on the MMI Runtime Framework proposal

Thirdly, the consistency of the system is reduced. Ad hoc solutions for these emerging requirements can increase complexity in the implementation process and reduce the pragmatical value of the MMI Framework & Architecture proposals.

For these reasons, the Soa2m architecture is an attempt to explore the possibility of some extensions of the current proposal, and this set of managers is the first step in this direction. [**Figure 2.1.11**]

94 The conceptual model behind the current recommendation follows the idea of an entity constructed in three levels perception (view), decision and knowledge (control) and memory (model) collaborating with a set of external entities and phenomena called the environment and the context. Recent proposals begin to show that perception, decision and knowledge depends on states (embodied, affective, social) related to a certain internal image of the environment, and memory structures are also «formatted» by the context. Thus, some functional blocks must be proposed to handle these internal images of external phenomena affecting the generation of goals, the performance of responses and interpretation of sensory experiences of the multimodal system.

2.1.1.4 Design Patterns of the Architecture

Ranges of complex implementations in Soa2m

The Soa2m proposal does not assume that all the multimodal systems are as complex as the architecture describes. Concrete implementations of multimodal systems may occur in a wide range of possibilities. It will depend on the chosen type of Modality Component, of Controller Component or Data Component.

As we already show[95], the MMI Framework & Architecture proposal allows two types of Modality Component: the Simple Modality Component and the Complex Modality component, that can include a nested Interaction Manager and a nested Data Component. [**Figure 2.1.12**]

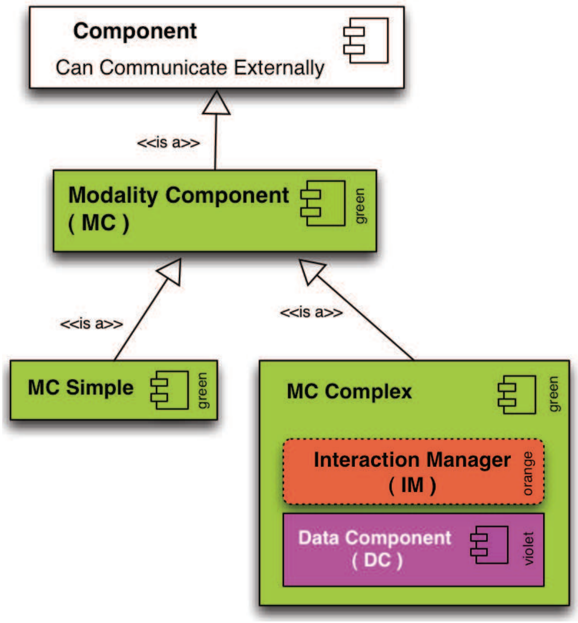


Figure 2.1.12: The Types of Modality Components in MMI

The Soa2m approach takes the same classification but the structure reflects the added Components and Managers. As a result, this increases the number of possible implementations.

Proposal: The forgetter and the chronicler controller design patterns

To begin, in the Soa2m architecture it is possible to implement two patterns of **Controller Components** (CC) depending on the needs of the *multimodal system*. **[Figure 2.1.13]**

The first pattern, is a Controller that does not need to have an explicit management of its current or past states, and does not need to estimate future outcomes. **[Figure 2.1.12]** It handles only the interaction from a turn taking point of view, and it does not need any context awareness.

This pattern of Controller is called the **Forgetter Controller Component** (CC Forgetter). **[Figure 2.1.13]** This implementation can be useful for systems with direct dialog strategies, controlled environments or a less amount of dynamic data.

Unlike the **CC Forgetter**, the **Chronicler Controller Component (CC Chronicler)**[96] keeps track of the state of the interaction cycle, the participants and the environment and context.

95 See supra section 1.3.1.1. The Interaction Model in the MMI Architecture

96 Forgetter, Chronicler, Dumb and Smart are multimodal design patterns provided by the Soa2m architecture with the goal of a better buildability(RI4), memorability, learnability, and satisfaction (RI2).

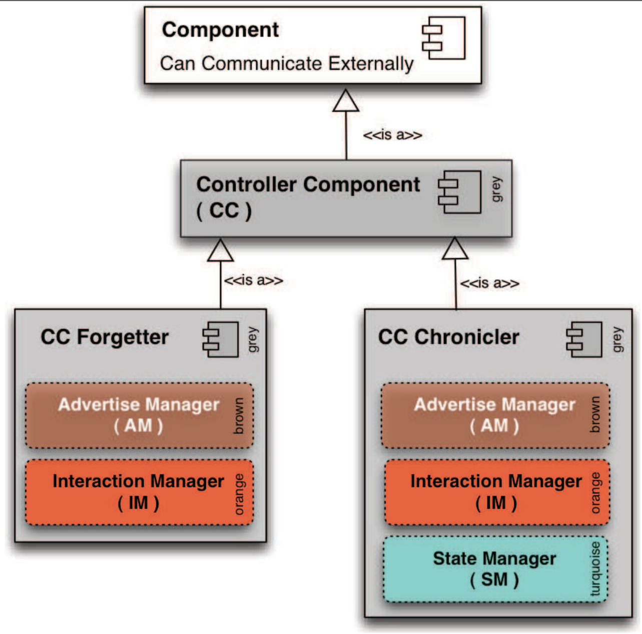


Figure 2.1.13: The Types of Controller Components in Soa2m

It can remember the past activities as episodes, is (at least) aware of some attributes of its current situation and eventually, it may project some outcomes in the future. [**Figure 2.1.13**] This implementation pattern can be useful for systems with a need of real-time sensing with incomplete and dynamic data, adaptive behavior, distribution and learning.

To continue, in the Soa2m architecture it is also possible to implement two patterns of **Data Components** (DC) depending on the needs of the *multimodal system*. [**Figure 2.1.14**]

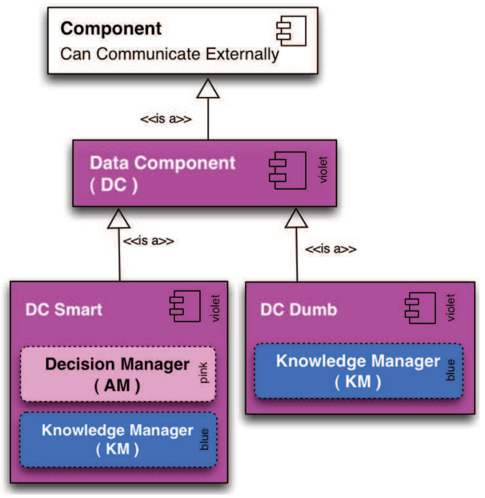


Figure 2.1.14: The Types of Data Component in Soa2m

The first pattern is the **Smart Data Component** (**DC Smart**), which is an implementation of the basic **Data Component** proposed by the MMI Framework & Architecture. It allows the multimodal system to make decisions based on some knowledge. In other words, it supports the decision making process enriched by the knowledge handling. As a result, the system becomes «smart» and in some extent, autonomous.

Proposal: the smart and dumb data component design pattern

This implementation pattern can be useful for systems with a need of autonomous and reactive behavior, learning and prediction.

The second pattern is the **Dumb Data Component (DC Dumb),** that inherits of some kind of knowledge (e.g. models) and provides it without any collection of dynamic data or decision making. In this case, the **Data Component** implementation is useful to give access to master data[97], leaving to the original MMI Framework & Architecture part, the storage of the transactional data produced by modules like the State Manager.

Finally, as we introduce in this part, the Modality Components in Soa2m are of the same types of the W3C's proposal but reflect the Soa2m structure: the **Simple Modality Component** (**MC Simple**) and the **Complex Modality Component** (**MC Complex**). [**Figure** 2.1.15]

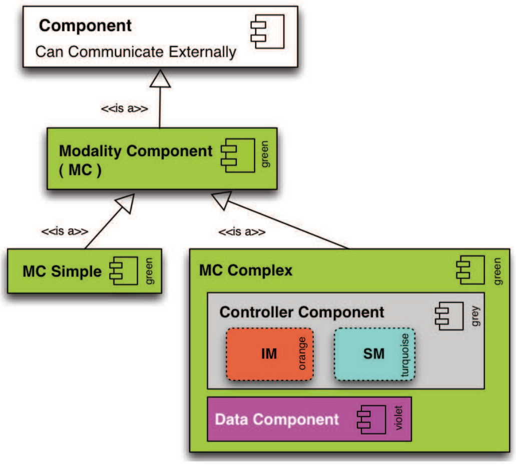


Figure 2.1.15: The Types of Modality Components in Soa2m

This taxonomy of Components can give multiple combinations in complex Modality Components; when we mix the characteristics described above to the imbrication features proposed by the Russian Doll principle of the MMI Framework & Architecture specification. Examples of the use of these implementation patterns are illustrated in [**Figure** 2.1.16].

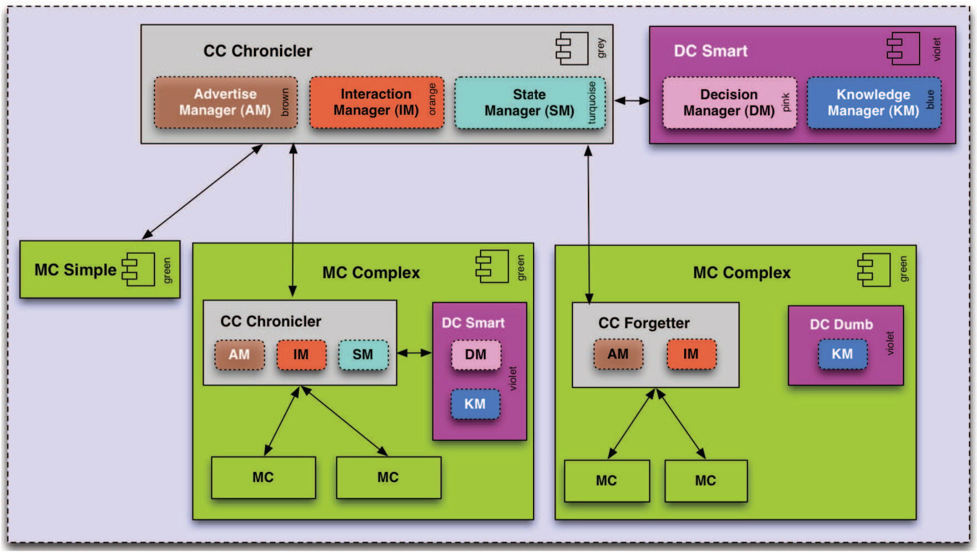


Figure 2.1.16: Complex Modality Component implementation in Soa2m

97 Master data includes the information that is important to a domain of application, in this case, the multimodal interaction handling and the application data. Transaction data are data that describes events that occur within the domain. Master data is less volatile than transactional data: entities and attributes in master data changes rarely. Master data is shared between different applications and generally requires to be stored in a different place than the transactional data, in order to protect it from the possible failures or attacks created by transactions.

Architectural issues addressed by Soa2m

In contrast, [**Figure** 2.1.17] shows the possible implementations of the MMI Framework & Architecture, and reflects some of the issues that we want to address.

First, the consistency (**RD8**) of the building blocks reflected in the naming convention and the functional distribution (e.g. managers are at the same level with components, why? in what aspect are they at the same level?), but also in the recursion process are an issue that we want to address, following the requirements of buildability (**RI4**), extensibility (**RD1**), integrability (**RI6**) and scalability (**RR5**).

Second, as we can see, the recursion does not follow the self-similarity (*mise-en-abîme*) criteria that the nested dolls principle announces. This can be an obstacle to the learnability (**RI2**).

For example, this is not clear if a Complex Modality Component with a Data Component inside can be implemented, and neither how the communication with a nested Modality Component could be possible.

Another example is how a Data Component could communicate with the MMI Framework components, like the Session Components, and if this communication must also pass through the Interaction Manager or not?

Third, the interfaces between the MMI Framework components and the Interaction Manager are not detailed.

It is not clear if the application must be developed as an Application Component or if the application component handles the relationship with an external entity that could be a client application. And how this interface could work? For us, this is an important point, because we want to address the emergent topic of user-interface delegation[98].

Other option could be that a Modality Component encapsulates the application logic. But in this case the consistency (**RD8**) of the structure can be questioned.

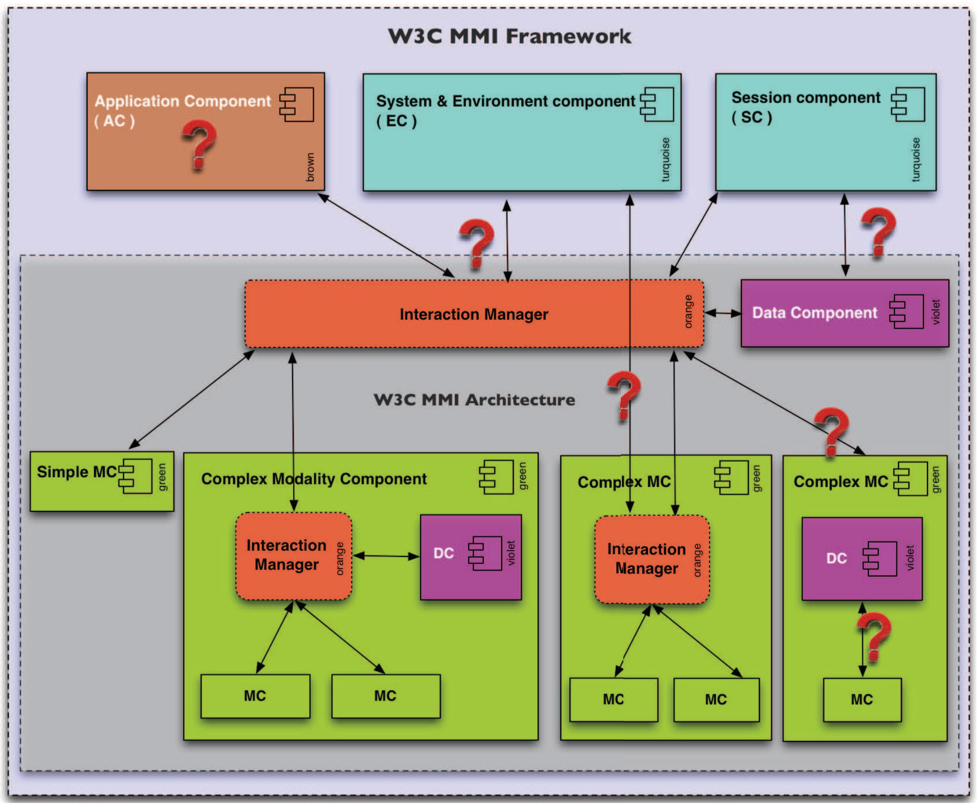


Figure 2.1.17: Possibles implementations of the MMI Architecture

98 See supra section 1.3.4.3 Summary

And finally, the interface between a nested Interaction Manager and the MMI Framework components is not defined, and as we already discuss, the Complex Modality Component could need some global information about the context or the session. In this cases, it must be necessary to demand these information to the parent Interaction Manager but this could finally mix a request for contextual data with the interaction control data. And a direct interface with these components, for example, with the System & Environment Component is not described as [**Figure 2.1.17**] shows.

The principal reason of all these unresolved questions is that the interfaces and connectors of the Data Component and the MMI Framework Components are not already detailed in the proposal. Only a generic recommendation is given.

In this sense, the Soa2m proposal of Components, Managers, Services and patterns is an extension and a possible contribution to the MMI Framework & Interfaces specification.

2.1.2 Soa2m Process View

In this section we will present the dynamic aspect of the Soa2m architecture, namely, how the MMModules communicate in run-time. At the same time it will introduce the communication model that defines how the MMModules[99] communicate in Soa2m. Other details about the process, like activities and component interaction, are for us, implementation-dependent. For this reason, these aspects will be detailed later, in section 2.3 The Soa2m Tools Implementation.

2.1.2.1 Communication Policy for Components

In [**Figure 2.1.18**] we resume the issues concerning the communication policy raised in section 2.1.1.3. Somme exchanges are explicitly allowed, others explicitly forbidden (**X**), others undefined (**?**) and others are mostly undefined but could be forbidden (**X?**).

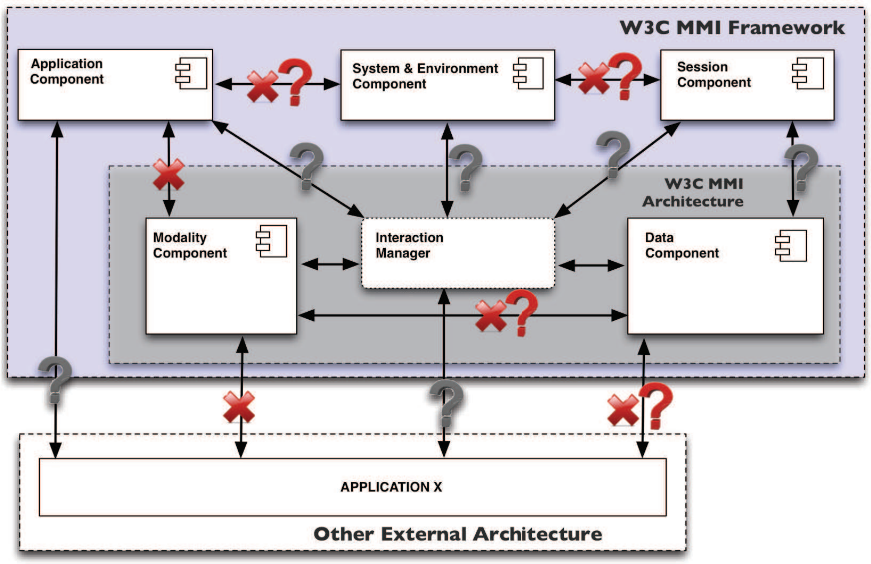


Figure 2.1.18: Communication Policy in the MMI Architecture

Justification of the MMI Communication Policy

The assumption that some exchanges could be forbidden in the communication policy is found on the history of the MMI Framework & Architecture proposal.

The MMI Architecture is inspired on the Galaxy Communicator Software Infrastructure (GSCI): a distributed, message-based, hub-and-spoke infrastructure optimized for constructing spoken dialogue systems. [Seneff et al. 1998]

99 For the MMModule definition, see supra section 2.1.1.1 The Basic Elements of the Soa2m architecture

This is a system architecture based on connections arranged like a chariot wheel, in which all traffic moves along spokes connected to the hub at the center. [**Figure 2.1.19**]

The GSCI architecture allowed to weave together servers which may not otherwise have been intended to work together, by rerouting messages and their responses and process results. The scripting capabilities of the hub allowed to insert simple tools and filters to convert data among formats, and modify the message flow of control in real time. It was also conceived to support the sharing of useful tools to other application participants.

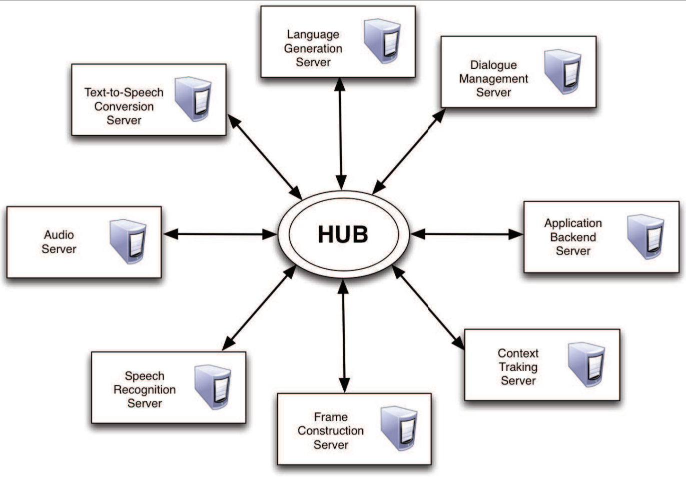

Language Generation Server
Text-to-Speech Conversion Server
Dialogue Management Server
Audio Server
HUB
Application Backend Server
Speech Recognition Server
Context Traking Server
Frame Construction Server

Figure 2.1.19: The Architecture of the Galaxy Communicator Software Infrastructure

And finally, the architectural pattern allowed the implementers to develop platform- and programming- language-independent services for recognition, synthesis, and other resources. For all these reasons, it was selected as a starting point to the MMI Recommendation.

However, keeping this in mind, we can suppose that the explicitly forbidden (**X**), undefined (**?**) and could be forbidden (**X?**) connections can be resolved [**Figure 2.1.17**].

We can assume that in the MMI Framework & Architecture specification (in its current state) all the components must communicate through the Interaction Manager, which is the «HUB» for every internal exchange and every external request, which is a potential bottleneck.

This assumption is made supposing that the generic approach drives to find some responses in the premises introduced in the recommendation. In this case, the generic proposal provides a tacit solution: the original intent of the proposal in respect of the consistency requirement (**RD8**).

Consequences of the communication policy

Then, the genericity and expressivity of the proposal can lead developers to a) seek an ad-hoc solution that could be contrary to the original structure of the proposal or b) follow the original solution, namely, the hub-and-spoke pattern design.

The balance between expressivity and completeness (**RD7**) and consistency (**RD8**) is an known issue in architecture design. In the case of the W3C proposal, the undefinition of some aspects of the MMI Framework & Architecture is an explicit bias that comes from its generic approach and is guided by the concern about extensibility (**RD1**), interoperability (**RR1**) and integrability (**RI6**).

Nevertheless, in reaction to this undefinition and the issues raised in [**Figure 2.1.16**] we propose an explicit communication policy for the Soa2m Architecture, which we expect to be less undefined, more consistent and yet, expressive. It is depicted in [**Figure 2.1.20**].

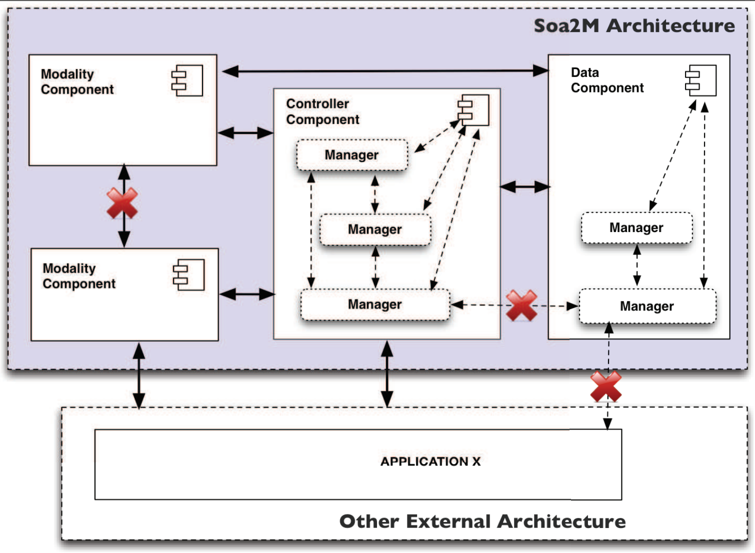


Figure 2.1.20: The Soa2m Communication Policy

P1: Controller Components can communicate with Modality Components, Data Components, internal Managers and with external entities.

P2: Modality Components can not communicate with each other.

P3: Modality Components can communicate with Controller Components, with Data Components and with external entities.

P4: Data Components can not communicate with each other.

P5: Data Components can communicate with Controller Components, Modality Components, and internal Managers.

P6: Managers can not communicate with external entities.

P7: Managers can not communicate with Managers contained in an entity different to its parent Component.

P8: A Manager can communicate with its parent Component.

P9: Managers can communicate with other Managers contained in the same parent Component.

This policy respects the basic communication guidelines given by the MMI Framework & Architecture with the **P1** and **P2** policies. In addition, we propose a set of policies that concern subsystems proposed by Soa2m: **P6**, **P7**, **P8** and **P9**.

Finally, the policy **P4** and **P5** is proposed to ensure the protection of data and to keep it as a back-end processing component, according to the security requirement (**RR6**).

Policy **P3** is introduced to allow external entities, like a client application delegating the management of its user interface, to control directly Modality Components through a dedicated interface.

In this way, this policy can authorize shortcut mechanisms like the ones implemented with the Function Modeling in SMARTKOM[100] and the Hardwired Reactions of REA[101].

100 See supra section 1.3.3.2 REA-1999
101 See supra section 1.3.3.4 SMARTKOM-2003

2.1.2.2 Communication Policy for Services

Finally, these communication policies presented in the last section are completed by communication policies for services in Soa2m [**Figure 2.1.21**]. This policy reflects the component's communications:

- **PS1**: Outgoing Services can communicate with its wrapped Component.
- **PS2**: Shy Services can communicate with its wrapped Manager.
- **PS3**: Outgoing Services can request other Outgoing Services.
- **PS4**: Shy Services can request other Shy Services in the same parent Component.
- **PS5**: Components can request Outgoing Services.
- **PS6**: Managers can request Shy Services.
- **PS7**: Components can not request Shy Services of other Components.
- **PS8**: Managers can not request Outgoing Services of other Components.
- **PS9**: Shy Services can request its parent Outgoing Service.

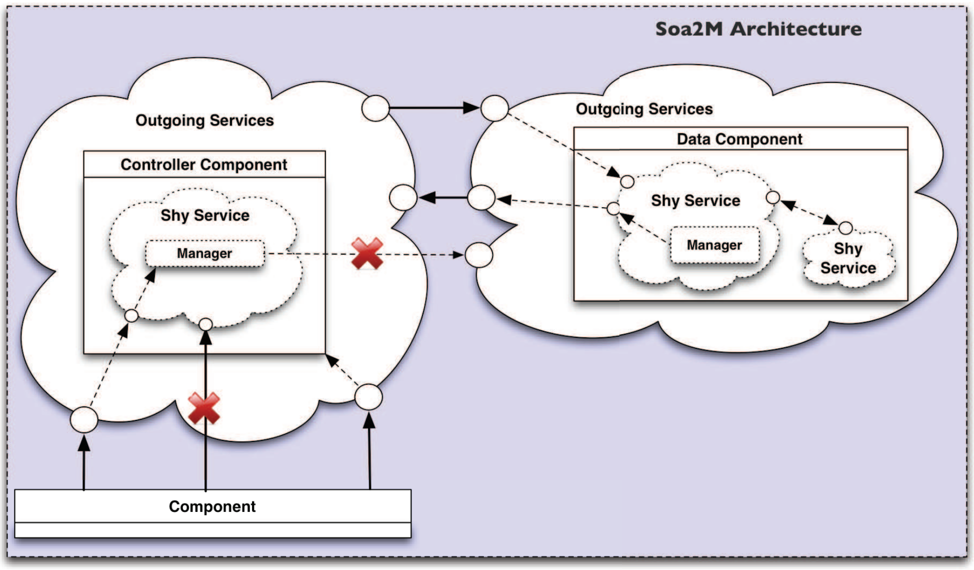


Figure 2.1.21: The Soa2m Services Communication Policy

2.1.3 Soa2m Implementation View

In this section we will present the implementation aspects of the Soa2m architecture. Outline First, we will present the mechanism proposed for the creation of components. Second, we will present how these components can be deployed. Finally, we will present the message model of Soa2m, which describes: how messages are transmitted and the structure of message; and the relationship between senders and receivers.

2.1.3.1 The Creation of Components in Soa2m

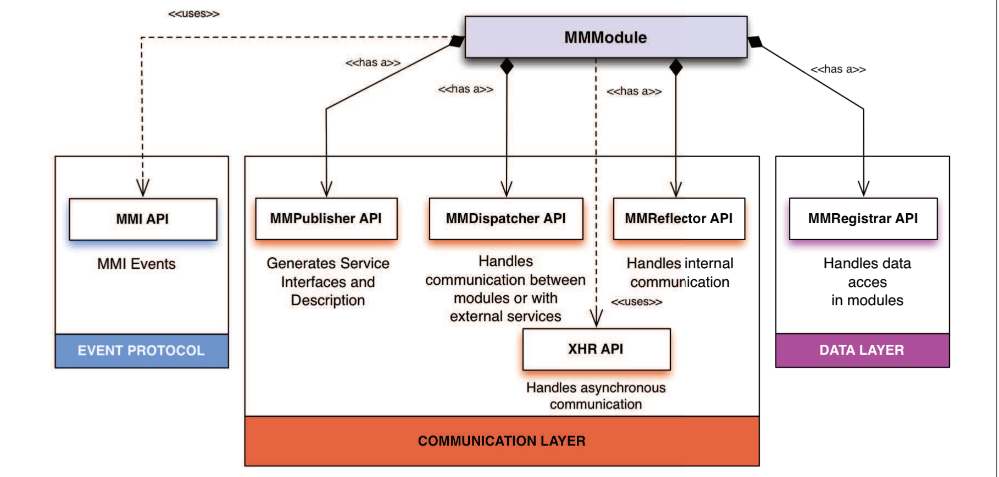
As we explained in section 2.1.1.1 Basic Elements of the Architecture the basic element in the Soa2m architecture is the MMModule. Each MMModule is implemented with a series of basic API's needed to communicate and to handle data. [**Figure 2.1.22**] These API's implement features that are common to all the modules in the architecture.

Basic APIs of the MMModules

The first API is the MMI API. This is the library that implements the Interaction Life-Cycle Events of the MMI Framework & Architecture recommendation.[32] [**Figure 2.1.22**] The second API, the MMPublisher, allows the invocation of services by generating dynamically its interfaces and description.

The MMDispatcher API, manages the asynchronous communication between modules. With this goal, it uses the XHR API, which is the library provided to send and receive the XML HTTP Requests.

The MMReflector API is responsible of the communication between Service interfaces and Components or Managers. It handles the internal (reflexive) communications. Finally, the MMRegistrar API, handles the access to the data, distributed internally or externally.



<<uses>>
MMModule
<<has a>>
<<has a>>
<<has a>>
<<has a>>
MMI API
MMI Events
EVENT PROTOCOL
MMPublisher API
Generates Service Interfaces and Description
MMDispatcher API
Handles communication between modules or with external services
MMReflector API
Handles internal communication
<<uses>>
XHR API
Handles asynchronous communication
COMMUNICATION LAYER
MMRegistrar API
Handles data acces in modules
DATA LAYER

Figure 2.1.22: The Basic APIs of the MMModules in Soa2m

From these basic modules the Soa2m architecture builds Components and Managers.

This process is designed following the abstract factory design pattern, suggested to create a family of objects without depending of its concrete classes. It define an abstract interface for creating an object, but let the classes that implement the interface decide which class to instantiate.

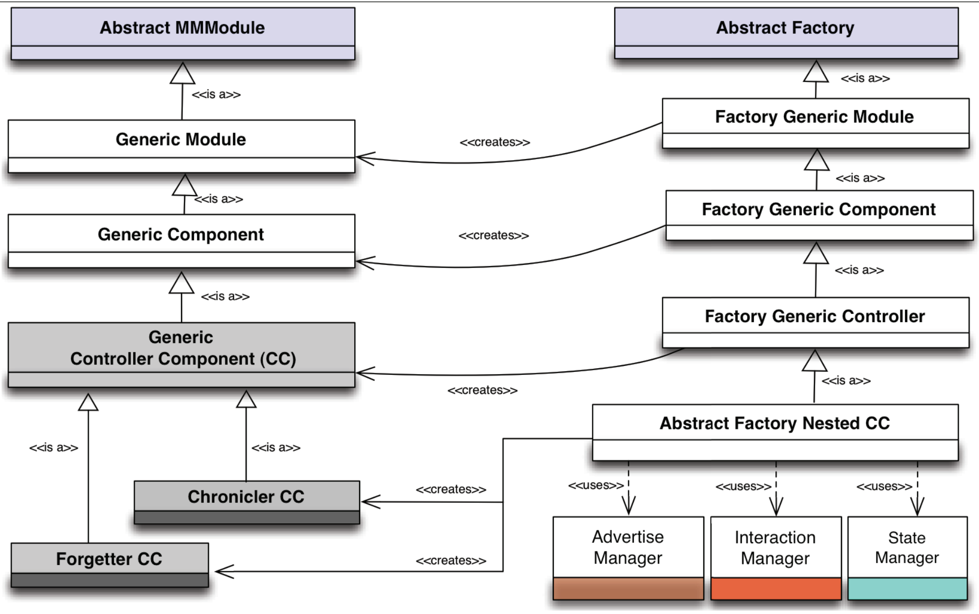


Figure 2.1.23: The Controller Component Factory

In Soa2m all the modules are produced according this design pattern, by classes responsible to the creation of different types of Components, Managers, Services and data structures. For example, in [**Figure 2.1.23**] is depicted the mechanism for the creation of a Controller Component in Soa2m, and the CC Component design patterns.[102]

By the use of the factory pattern we can ensure flexibility (**RD2**), buildability(**RI4**), configurability (**RI3**), extensibility (**RD1**) and modifiability (**RD5**) in the creation of components. The reason is that using factories allows the inclusion of new Soa2m design patterns in the future to the Components, Managers and Services. For example, using the same mechanism (and great number of common classes) [**Figure 2.1.24**] we can produce a **Data Component** and its design patterns:

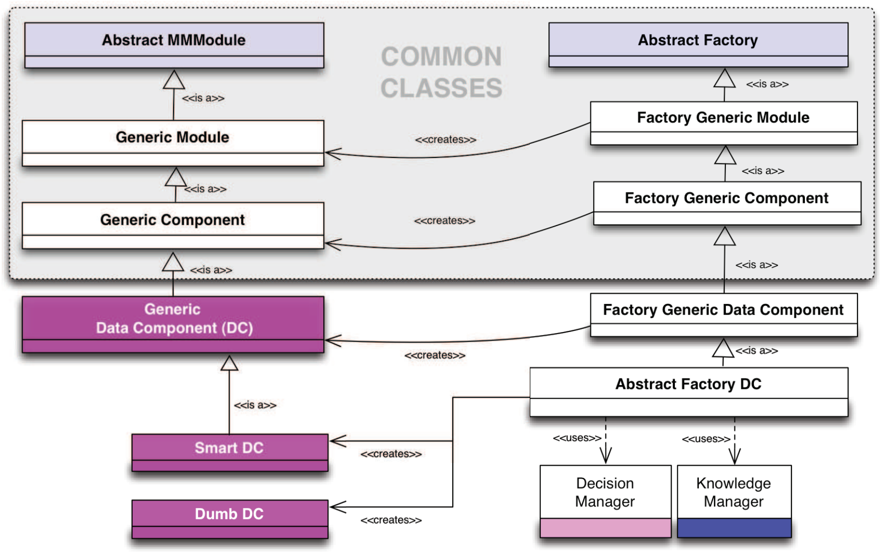
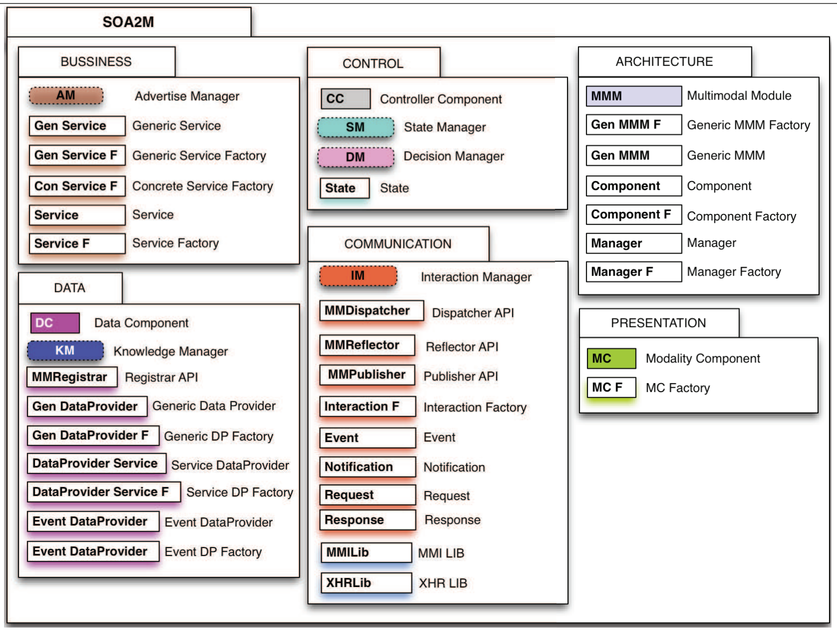


Figure 2.1.24: The Data Component Factory

As a result of this bias, some classes are provided for the creation of the modules as shows the implementation view of the Soa2m packages in [**Figure 2.1.25**].



SOA2M
BUSSINESS
AM Advertise Manager
Gen Service Generic Service
Gen Service F Generic Service Factory
Con Service F Concrete Service Factory
Service Service
Service F Service Factory
DATA
DC Data Component
KM Knowledge Manager
MMRegistrar Registrar API
Gen DataProvider Generic Data Provider
Gen DataProvider F Generic DP Factory
DataProvider Service Service DataProvider
DataProvider Service F Service DP Factory
Event DataProvider Event DataProvider
Event DataProvider Event DP Factory
CONTROL
CC Controller Component
SM State Manager
DM Decision Manager
State State
COMMUNICATION
IM Interaction Manager
MMDispatcher Dispatcher API
MMReflector Reflector API
MMPublisher Publisher API
Interaction F Interaction Factory
Event Event
Notification Notification
Request Request
Response Response
MMILib MMI LIB
XHRLib XHR LIB
ARCHITECTURE
MMM Multimodal Module
Gen MMM F Generic MMM Factory
Gen MMM Generic MMM
Component Component
Component F Component Factory
Manager Manager
Manager F Manager Factory
PRESENTATION
MC Modality Component
MC F MC Factory

Figure 2.1.25: The Packages of Soa2m

102 See supra section 2.1.1.4 Design Patterns of the Architecture

2.1.3.2 The Deployment of Components in Soa2m

Client-Server implementation of the Soa2m architecture

Soa2m is implemented according a multi-tiered architecture. The data access (the Data Component) and the compute-intensive portions (the Controller Component) are included in a web server (apache) while the Modality Components can be distributed in the form of Http Clients in a browser or in stand-alone applications..

The technologies available for the implementation are PHP and MySQL on the server side and HTML, Css, Svg, EcmaScript, ActionScript, Flex, Flash, Air, Gpac and Java on the client side.

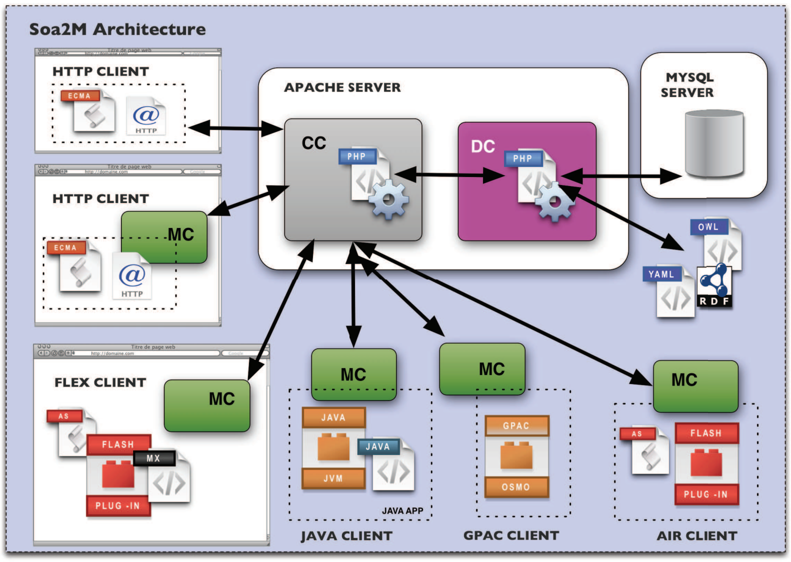


Figure 2.1.26: The distribution of Components in Soa2m

2.1.3.3 Structure and Transmission of Messages in Soa2m

The implementation choice presented in the last section reflects the message model of Soa2m, which describes: how messages are transmitted and the relationship between message senders.

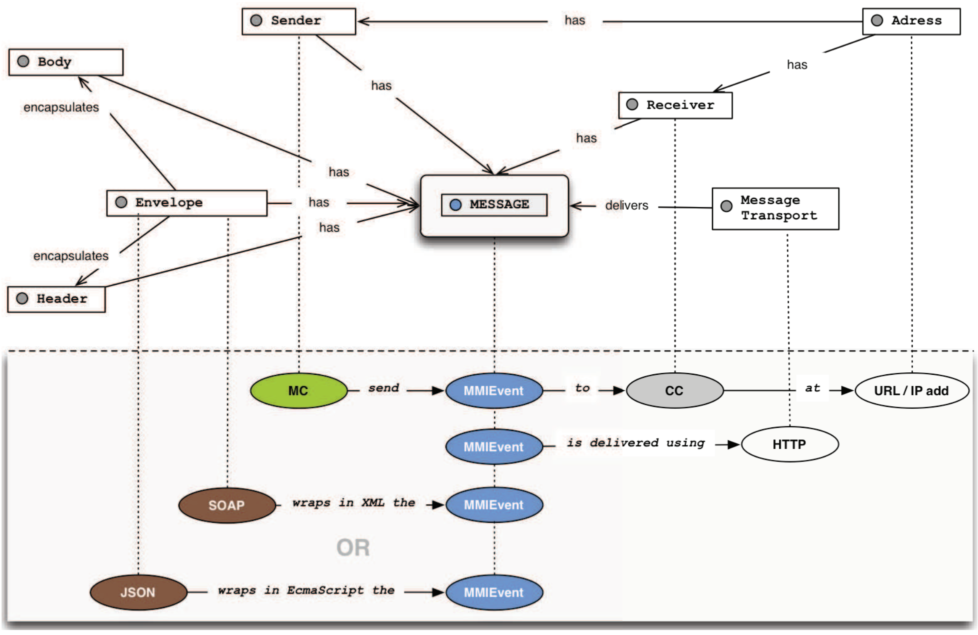


Figure 2.1.27: Message Model in Soa2m: Message Transmission

On one hand, Soa2m messages (MMI Events) are transmitted [**Figure 2.1.27**] following the recommendation MMI Framework & Architecture, namely, sending an XML message using the Http transport layer.

To this, we add the service-oriented perspective, in order to support the abstract **layer 7: Application Logic**[103] and more precisely, to support the communication with the Advertise Manager and its services.

With this goal, the MMI Events are enveloped following the SOAP (Simple Object Access Protocol) protocol or using the structure proposed by the SOAP envelope but using another document format, the JavaScript Object Notation (JSON)[104]. [**Figure 2.1.27**]

SOAP in Soa2m

A SOAP [W3C-SOAP 2000] message encapsulates all the necessary information required by the service provider to carry out the action in an envelope divided in two parts, an optional header and a mandatory Body. [**Figure 2.1.27**]

For example, if a Modality Component service description states that a TTS process requires a Privacy variable as input, the SOAP message will contain such a variable, along with a desired value (e.g. Privacy = 'Intimate').

On receiving a SOAP message, the service provider carries out the action, and may return a different SOAP message which encapsulates any output from the service (such as confirmation variable or an error).

Using SOAP allows service invocation to be independent of any particular platform or protocol that the service provider is using; and to add semantic technologies to the message to enhance the discovery and register of the processes offered by the service.

For its richness in description, automation and process handling, SOAP is a good choice when the resource to access is a process, and this process will collaborate with the processes on the client requesting the service. Thus, a SOAP service affect the «state» of the requesting client processes.

REST in Soa2m

In contrast, the Representation State Transfer (REST) approach is a good choice when the resource to access is a known data repository or data structure.

The REST interactions are «stateless» in the sense that the meaning of a message does not depend on the state of the exchange. Instead of wrapping many operations in one single service, it exposes an URI with just 4 standards operations (get, post, put, and delete).

For REST compliant web-services, JSON [IETF-RFC4627 2006] is an alternative to the XML format of request/response documents, mostly used in web-services for serializing and transmitting structured data over a network connection with the EcmaScript [ISO-16262 1997] (or Javascript) language.

The message data structured with the XML format is:

```
<a:Contact><a:FirstName>Jhon</a:FirstName><a:LastName>DOE</a:LastName></a:Contact>
```

While the message data structured in the JSON format is :

```
{"Contact":{"FirstName":"Jhon","LastName":"DOE"}}
```

As you can see, JSON is a more compact format, that reduces the network traffic 15862 bytes for a JSON request while 31872 for SOAP for the same data content.[105] In contrast, the time needed for consuming a JSON message is higher that a SOAP message: to treat 3000 requests, JSON takes 14 seconds while 10 seconds for SOAP.

Therefore, to respond to the requirements of completeness (**RD7**), extensibility (**RD1**), integrability (**RI6**), interoperability (**RR1**) but also with concern in performance (**RR2**) and availability (**RR4**), the Soa2m architecture support both message formats as showed in the message model in [**Figure 2.1.27**].

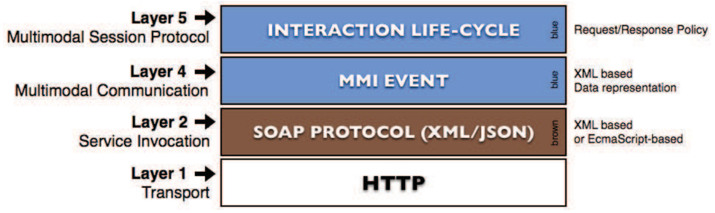
103 See Supra section 2.1.1.2 Abstract Layers of the Architecture

104 More details about the format of messages will be given in part 2.3 The Soa2m Tools Implementation

105 Comparison available at: http://tinyurl.com/9l5fqtg

2.1.3.4 Messaging Protocol in Soa2m

On the other hand, the relationship between message senders and receivers is given by the Interaction Life-Cycle Events protocol. [**Figure 2.1.28**]



Layer 5 → Multimodal Session Protocol
INTERACTION LIFE-CYCLE
blue
Request/Response Policy
Layer 4 → Multimodal Communication
MMI EVENT
blue
XML based Data representation
Layer 2 → Service Invocation
SOAP PROTOCOL (XML/JSON)
brown
XML based or EcmaScript-based
Layer 1 → Transport
HTTP

Figure 2.1.28: The Life-Cycle Events and the Transport Layer

This protocol describes the exchange policy of MMIEvents. For example, to initiate a multimodal context (corresponding a multimodal session) three MMIEvents can be used: the NewContext, Prepare and Start Events. [**Figure 2.1.29**]

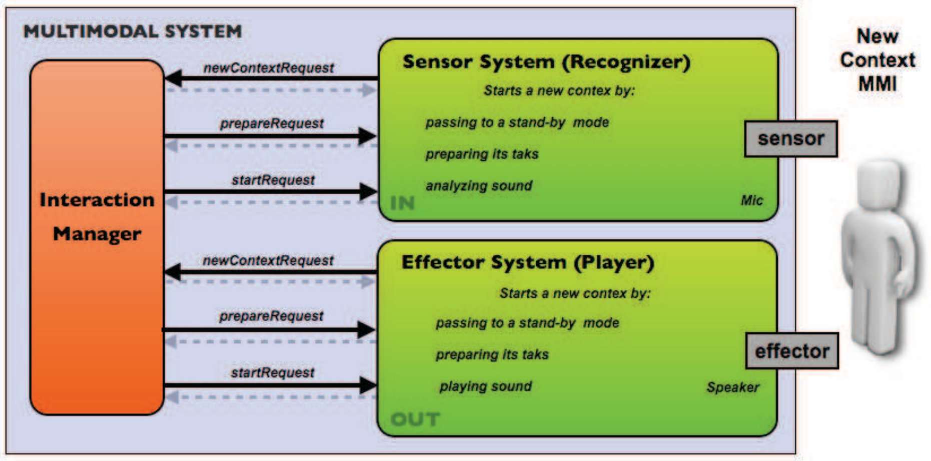


Figure 2.1.29: The NewContext triggering policy

The NewContext Event, must be triggered only by Modality Components and send to the Interaction Manager, if the response provides a new context, then the Modality Component can pass, e.g. to a stand-by mode: the multimodal session is then launched and waits to an interaction cycle to start. [**Figure 2.1.30**]

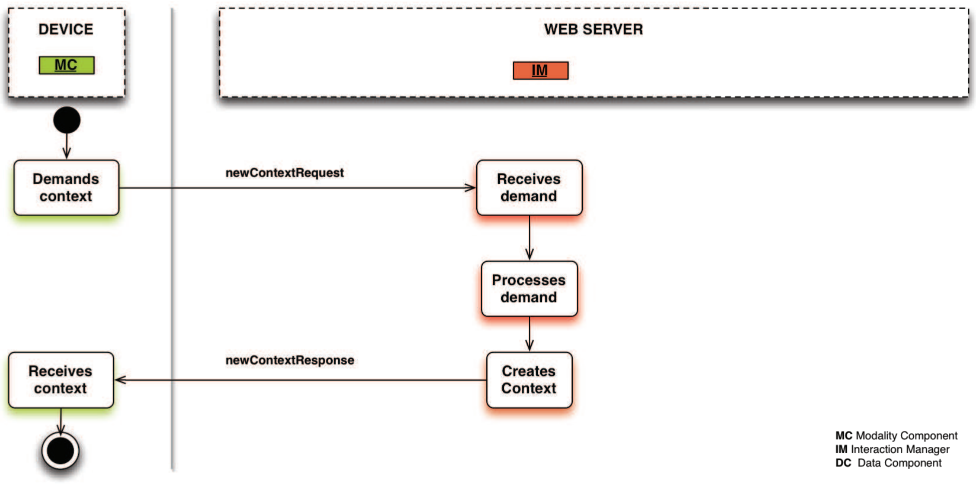


Figure 2.1.30: Creating a NewContext of interaction in MMI

The same policy for the context creation is used in the Soa2m architecture but as we already explained, the distribution of responsibilities changes and the Interaction Manager is limited to handle the interaction cycle and responsible of the integration and composition of modalities.

[**Figure** 2.1.31] shows the creation of a context in Soa2m with this perspective.

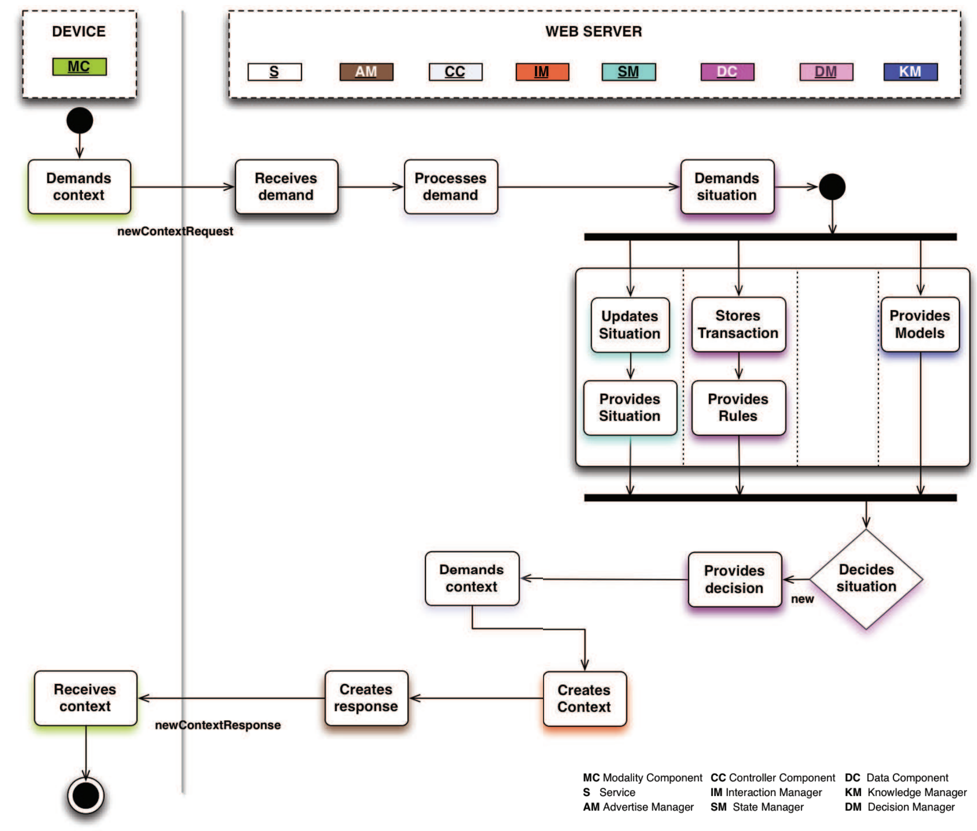


Figure 2.1.31: Creating a multimodal session in Soa2m

In contrast, the Prepare and Start Events must be triggered only by the Interaction Manager and send to Modality Components.

As a result, a Modality Component can not send a StartRequest or a PrepareRequest to the Interaction Manager. In both cases the Modality Component depends on the Interaction Manager to start an interaction. In consequence, the preparation of media or the start of the interaction cycle implies also the beginning of a multimodal session, as illustrates [**Figure** 2.1.32].

This policy of the Interaction Life-Cycle protocol is originated on the original goal of the protocol: this is a coordination mechanism, used to handle the turn-taking between multiple modalities.

The MMI Interaction Life-Cycle is a context-aware protocol, provided to support the interruption and adaptation of the interaction cycle to the situation of the user and the situation of the *multimodal system*.

The communication protocol in the MMI

For example, when the interaction cycle is launched by an user input (let's suppose that the context already exists) two MMIEvents must be used: the Extension Notification and the Start Event. [**Figure** 2.1.33]

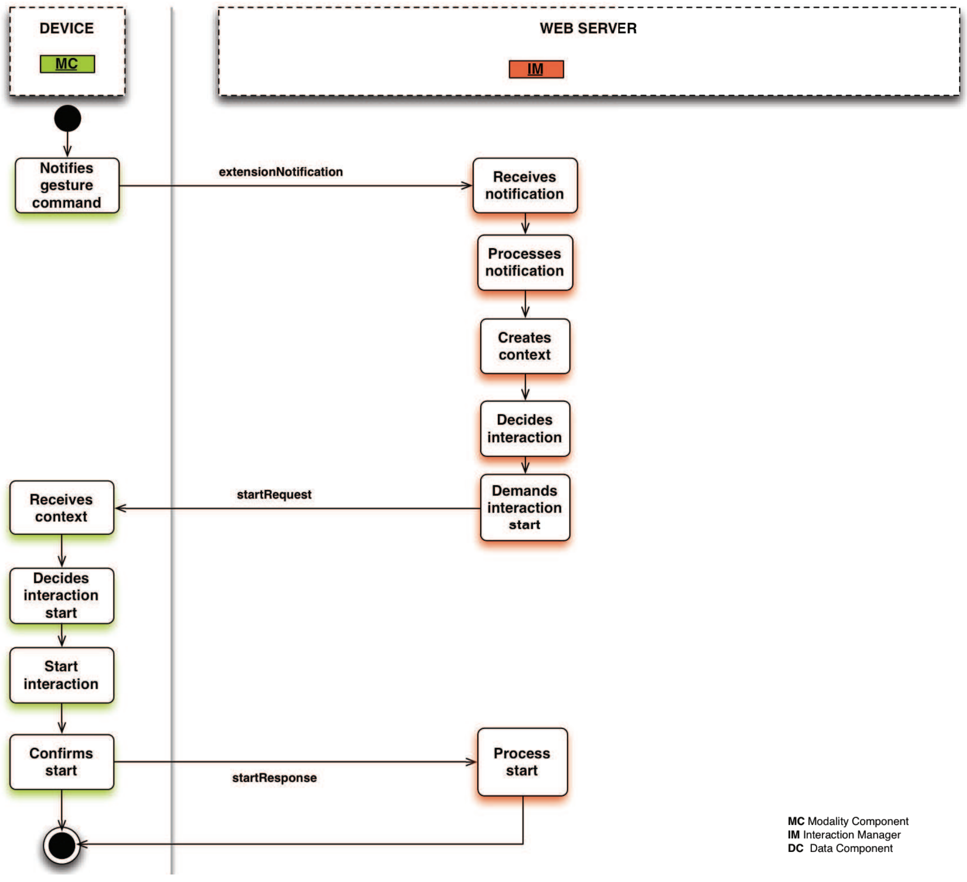


Figure 2.1.32: Creating a multimodal session with a Start Event MMI

When the *sensor system* captures a command (for example a voice command), [**Figure** 2.1.33] the Modality Component must send an Extension Notification (as we already show, it can not send Start events). Then, after processing the interaction situation, the Interaction Manager must send a StartRequest. If the Modality Component is still capable to start the interaction cycle, it launches the interaction and then, it sends a StartResponse with a status «true».

This policy shows how the evolution of the interaction cycle and its media (in this case, the presentation) will depend on the decisions at the control level and in the Interaction Manager of the *multimodal system*.

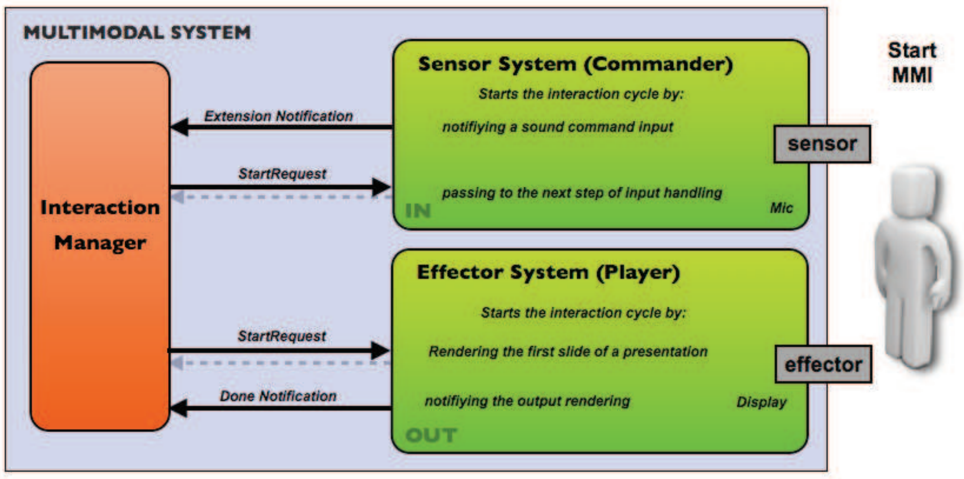

MULTIMODAL SYSTEM
Interaction Manager
Extension Notification
StartRequest
Sensor System (Commander)
Starts the interaction cycle by:
notifiying a sound command input
passing to the next step of input handling
IN
Mic
sensor
Start MMI
StartRequest
Done Notification
Effector System (Player)
Starts the interaction cycle by:
Rendering the first slide of a presentation
notifiying the output rendering
Display
OUT
effector

Figure 2.1.33: The Start triggering policy

In consequence, control events are originated in the Interaction Manager, and executed by the Modality Components. Notifications are the tool used to inform the changes on the user interface state like a discrete event (e.g. user input), the end of a continuous event (e.g. a recognition process or a sound reproduction) or any other change affecting the interaction.

[**Figure 2.1.34**] shows a generalization of this mechanism for the events controlling the input interaction, namely, the Start, Cancel, Resume, Pause, Resume and Prepare event on the input direction.

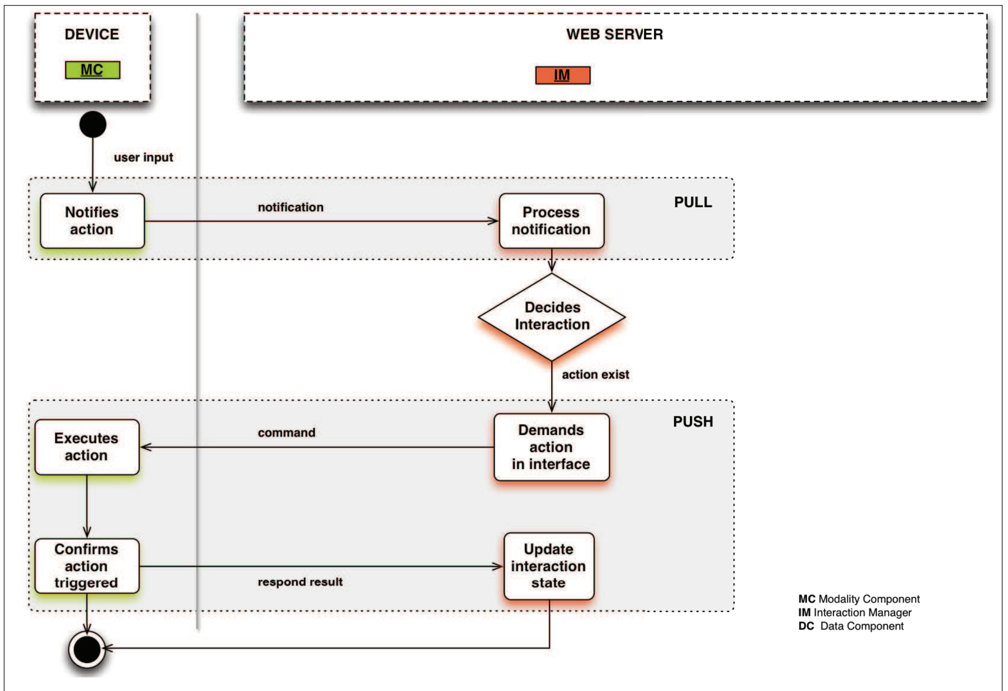


Figure 2.1.34: Generalization of the policy for input control events in MMI

For the output direction, the generalization of the same mechanism with the same events is provided in [**Figure 2.1.35**].

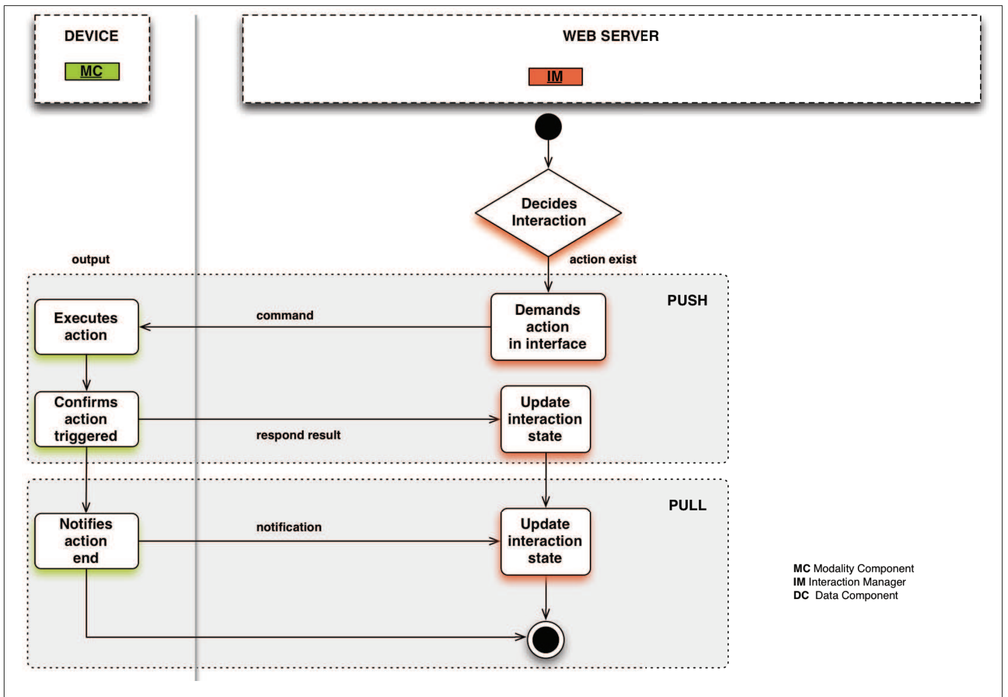


Figure 2.1.35: Generalization of the policy for output control events in MMI

As [**Figure 2.1.34**] and [**Figure 2.1.35**] illustrate, according to the MMI Framework & Architecture protocol, the command of Modality Components is initiated by the server, which means that if there is a client-server implementation, it must be designed following a push notification technique.

The push communication in the MMI specification

Push techniques allow the server to send new events or data to the client through progressive download or long-polling HTTP request from the client. Standards like MPEG-4 BIFS, Flash XML Sockets [ADOBE-Flash 2012] and HTML5 [W3C-HTML5 2012] with the EventSource interface, have the ability to define data streams used by the server to send commands, notifications or data updates to the client.

Push techniques enable client applications to subscribe to particular events, notifications or data streams and provide the server a callback address or a client-side service to which they are delivered.

In consequence, to implement a push technique, the server must know the client address in order to deliver the data, and some registry of registered subscribers must be created.

The server determines when things change, and simply sends down the new data. In this way, new HTTP connections don't have to be opened all the time. The downside is that the single connection between server and client, consumes a resource on the server side while it's open and the more push connections there are, the more it increases network traffic.

In the MMI Framework & Architecture protocol, the push technique is suggested but the implementation of this communication mechanism is not currently described.

Proposal: an adaptive pull communication for Soa2m architectures

Push techniques allow the server to send new events or data to the client through progressive download or long-polling HTTP request from the client. Standards like MPEG-4 BIFS, Flash XML Sockets [ADOBE-Flash 2012] and HTML5 [W3C-HTML5 2012] with the EventSource interface, have the ability to define data streams used by the server to send commands, notifications or data updates to the client.

For this reason, the Soa2m architecture, provides a proposal that encompasses the discovery & register of Modality Components[106] and an extension to the MMI communication protocol to cover the push and pull mechanisms. With this proposal, the Soa2m architecture responds to the requirements of completeness (**RD7**), extensibility (**RD1**), integrability (**RI6**) and interoperability (**RR1**) concerning the relations allowed between requesters and providers of messages.

Thus, in the Soa2m architecture, the communication protocol has been extended with an adaptive pull technique.

Pull techniques allow the client to request new data from the server; using forms submissions or AJAX-based technologies using the XMLHttpRequest object.

With this technique the change is instigated from the Modality Component. After a certain period, the client requests the server, who notifies the changes on the user interface or in the data, causing the client state to evolve, for example, by changing the user interface.

The connection is closed after each transfer and the client is told when to open a new connection, and what data to fetch when it does so. A pull technique is the best option for tightly coupled clients to which the server has reliable access while a push technique provide a small communication latency.

An important element to take into account is the pulling frequency: the time to update the user interface. High pulling frequency may induce redundant checks leading also to high network traffic. Low pulling frequency, on the other hand, may lead to missed updates.

As a result, ideally, the pulling interval should be equal to the rate at which the server state changes, thus, it must be adaptable and eventually handled by a context-aware system because the success of a pull-based technique hinges on the accurate estimation of the update interval value.

106 Using the Advertise Manager, the management of the state of the system by the State Manager and a register located in the Data Component. This mechanism will be described in detail in section 2.2. A Semantic SOA Architecture

In Soa2m this interval is calculated at the control level, namely in the Controller Component, using the contextual information provided by the State Manager and the Decisional processes provided by the Data Component.

First, a dedicated data structure is proposed: the timeout attribute. In Soa2m, a timeout is a tuple, an ordered list of three elements:

timeout ::= ⟨ ***S*** **,** ***L*****,** ***I*** ⟩

Where

S is the communication sleep period beginning on the communication timestamp
L is the communication validity limit, the life-time of the communication
I is the communication interval

Definition 2.1.1: The timeout triplet in Soa2m .

Proposal: the timeout triplet in Soa2m

In a distributed *multimodal system*, Modality Components can be idle for long time if no interaction cycle events happens or the situation is not optimal to allow a specific type of interaction. Given the fact that the data rate is very low during this period, it is not necessary to keep the client requesting all the time. The sleep value reduces the requesting time by putting the client(e.g. a Modality Component) into periodic sleep state.

Each client (e.g. a Modality Component) sleeps for some time, and then wakes up and checks to see if there are changes planned on the server side (e.g. The Controller Component). During sleeping, the client turns off update requests, and sets a timer to awake itself later. The sleep value is calculated by the Controller Component on the server side based on the context-awareness level of the *multimodal system*. It can be static and defined with a set of basic rules or more dynamic, linked to the semantic analysis of situations.

The second element of the timeout tuple is the communication life-time. A client leaves the *multimodal system* when its life-time is exceeded and needs to restart its registering mechanism to obtain a new client id and timeout. This allows to periodically update the metadata about the client and verify its validity (e.g. authorization).

The third element is the communication interval, which is modulated according to the *multimodal system* and server needs by a set of static rules or by a prediction mechanism used in The Controller Component. This element informs before-hand the client about the frequency of request that can be allowed by the server in the current conditions. This value is exchanged on each request, which means that it can be changed at any moment.

Proposal: two new events to handle discovery and registration in Soa2m

The timeout triplet, can be added in the data attribute of any of the Interaction Life-Cycle Events or in two dedicated events that the Soa2m architecture proposes: the checkUpdate Event or the UIUpdate Notification. [**Figure 2.1.36**].

The checkUpdate Event is provided to verify if there are any changes in the server side, to recover the eventual message (under the form of a MMI event) and to adapt the request timeout if needed, and to trigger automatic notifications about the state of the client (e.g. the Modality Component) if the automaticUpdate field in the response is true. [**Figure 2.1.36**]

The UIUpdate Notification is proposed to periodically inform about the state of the Modality Component, to help in the decision making process used by the Controller Component. [**Figure 2.1.36**]

The inclusion of the pull technique to the multimodal communication is founded in the nature of the data to exchange which is mostly discrete messages containing control data. As the comparative study of [Bozdag et al. 2006] concludes, push techniques brings scalability issues: the server application CPU usage is 7 times higher as in pull, and the server starts to saturate at 350-500 clients. In contrast, they ensure high data coherence and high network performance because it avoids unnecessary requests. This attributes are important when the goal is to exchange important amounts of data without losses.

On the other hand, pull techniques demand the adaptation of requests to the frequency of server updates.

If the communication interval is higher than the update interval, some data miss will occur and for example, the synchronization between the interaction cycle state on the client and the server can be loss. If it is lower, network performance will suffer.

With our proposal, the pull interval will equals to the publish interval because the client know the exact publish interval beforehand according to a time pattern. In this way, data coherence is ensured (**RD8**) and network performance maintained (**RR2**). Since the Controller Component in the server has access to all the state data, it can use a prediction algorithm implemented in the Data Component (in its Decision Manager) to foresee a time when the data is going to change. The server then attaches this time value in the timeout triplet to the outgoing data.

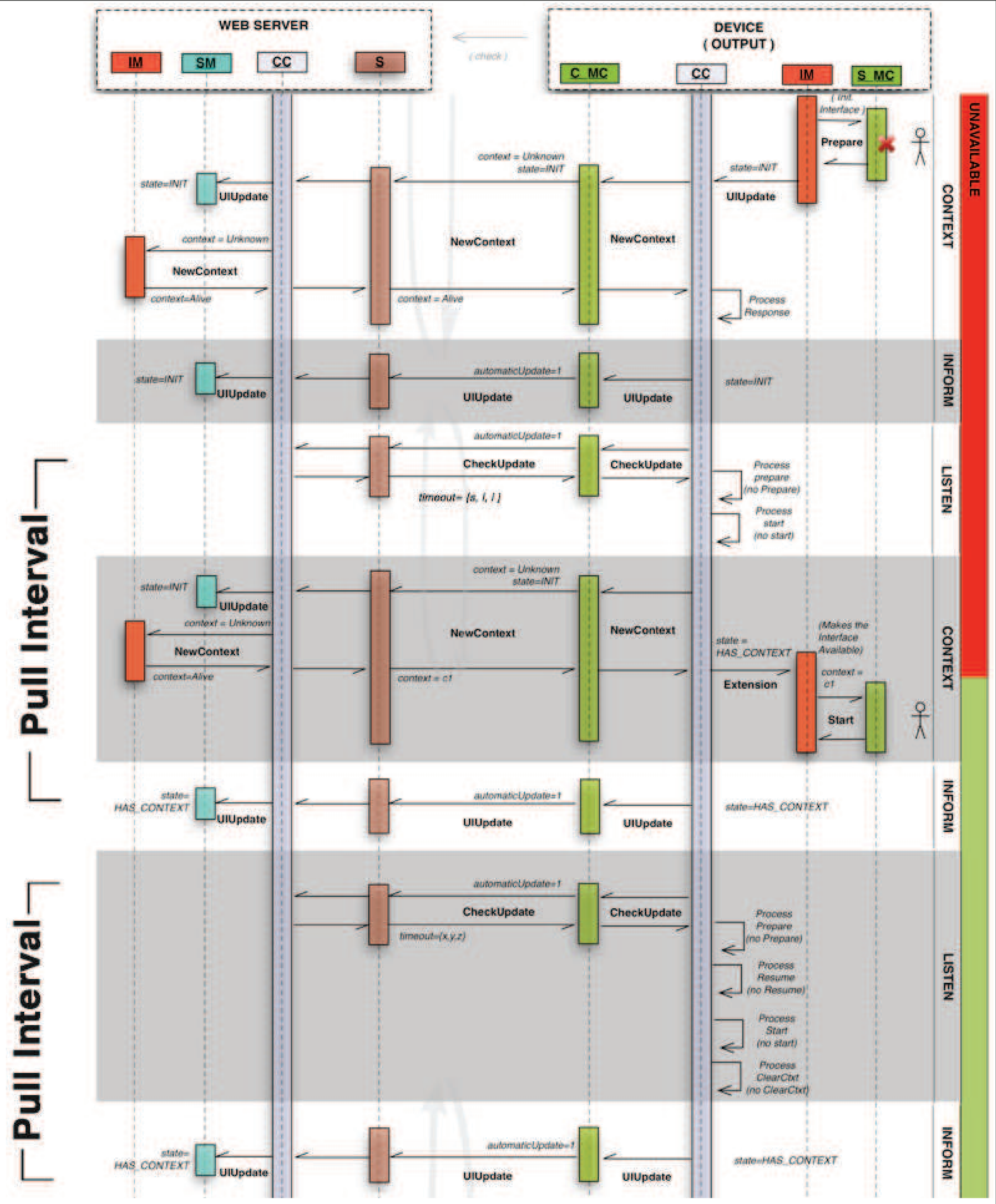


Figure 2.1.36: The pull mechanism using adaptive timeout proposed in Soa2m

Our approach needs a trace mechanism for the relevant data concerning the state to be handled on the server side. Thus, this is one of the responsibilities of the State Manager.

Finally, if the server prediction is wrong and still a change of interest occurs in data, if the server knows the address of the client it can push the change to it, using the original push technique proposed by the MMI Framework & Architecture proposal. In this case the push command is handled as an interruption of the default pull update mechanism. In this way, the system maintains its reliability (**RR3**).

Conclusion

To sum up, the relationship between message senders and receivers depends on the Interaction Life-Cycle Events protocol in the Soa2m architecture for a push implementation to which the Soa2m architecture adds a pull implementation based on a timeout data structure and two new events: the checkUpdate Event and the UIUpdate Notification.[37]

2.1.4 Soa2m Physical View

The MMI Framework & Architecture physical view shows how data can be organized and structured on physical devices. This is a distributed architecture, where the client application can be fully deployed in a local network, or it can be deployed in a large-network like the internet. [**Figure 2.1.37**]

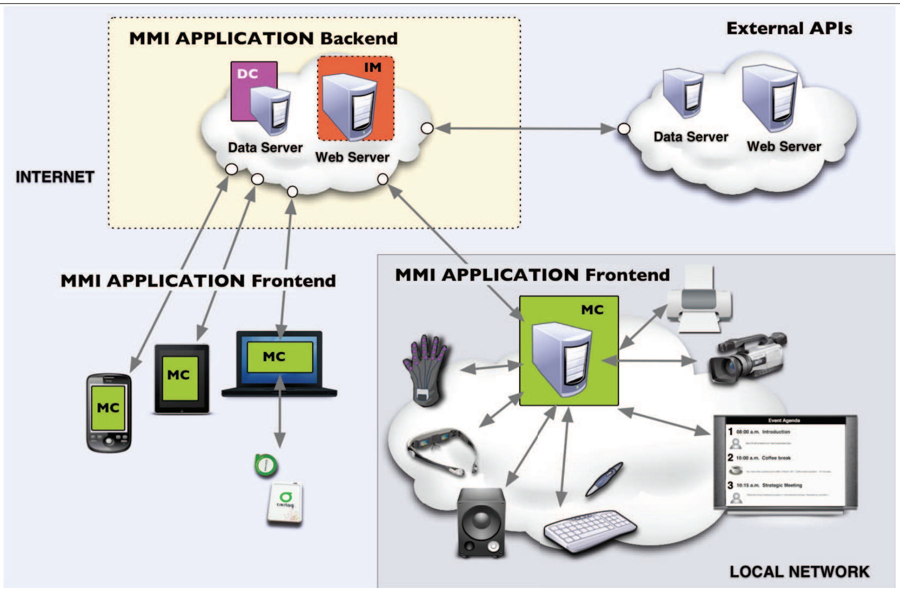


Figure 2.1.37: Example of deployment of an application using the MMI Architecture

Proposal: multimodal remote delegation in Soa2m

In contrast, the Soa2m architecture is deployed according a delegation principle (one of the emergent topics detected in section 1.3.4.3 Summary) in which the client application delegates to the multimodal system the management of the composition of multimodal services. This is called in Soa2m «the management of the remote user interface.» [**Figure 2.1.38**]

As we explained before, this remote multimodal interface can be handled by pushing business data to the Modality Components or by an adaptive pull mechanism implemented in the Modality Components to recover the business data.

The Soa2m services propagates push commands to registered Modality Component enabled to receive those commands. On one hand, each Modality Component establishes an accredited and IP connection with the service and receives Soa2m events[107] and business data over this persistent connection. This is the mechanism in the case of an implementation using push techniques.

107 The Soa2m Events are the MMI Life-Cycle Events and two proposed events: the checkUpdate Event and the UIUpdate notification.

On the other hand, each Modality Component can «check» for updates on Soa2m events and business data using adaptive pull techniques. The client application's backend can connect with the Soa2m services while monitoring incoming data intended for their application front-end. When new data for the application front-end arrives, the application's backend prepares and sends an event through the channel (or sends a pull request) with the intended data to the Soa2m services. Afterwards, the Soa2m CC services pushes (or publishes) the data to the target Modality Components after processing the multimodal *fission* according with the interaction context and the environment.

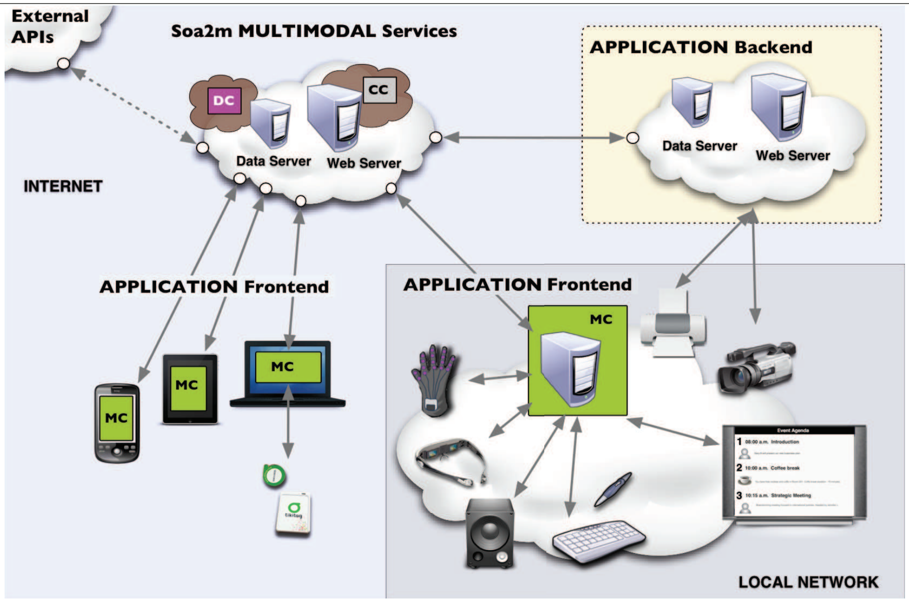


Figure 2.1.38: Example of deployment of an application using the Soa2m Architecture

The Soa2m CC services are the centerpiece of the remote interface management. They are provided for propagating interaction commands and business data updates to Modality Components hosted in devices such as smart-phones, tablets, laptops and ambient computing servers. [**Figure 2.1.38**]

Each Modality Component hosted in the device can send requests to its parent Outgoing Services or establishes an accredited IP connection with the parent Outgoing Services and receives commands and data over this persistent connection. Application's backend originate updates of the business data and the behavior guidelines of the interaction cycle in its server software. The flow of remote user interface data is two-ways. The application's backend composes a data package that includes a token for the client application (to cover the security concerns-**RR6**), a timeout and the payload.

The application's backend sends the data package to the Soa2m service which in turn, updates the multimodal behavior instructions (if needed) and/or pushes (or publishes) the data to the Modality Component.

Finally, the Soa2m service informs the application's backend about the multimodal session state through a feedback service that the provider connects with. The feedback service provides a list of Modality Components per application that were recent participants in a remote multimodal interaction cycle and their *fusion / fission* final results on the form of input>output pipelines that permit to handle the bidirectional nature of the system and its symmetry (**RF2**).

2.1.5 Conclusion

This section presented the Soa2m architectural approach, designed to address the working hypothesis that it could be possible to build a system bringing services to each user with the use of the semantically best resources available, independently of the communication channel, the *mode* of restitution, the *media* or the device.

After the selection of a reference architecture as a starting point, namely, the MMI Framework & Architecture Specification, the analysis of this standardized solution lead us to propose some extensions required to address our need.

A proposal presented in four views

This section presents these extensions from four points of view: a logic view of the proposal, a process view of the proposal, its implementation point of view and finally the physical view of the proposed deployment.

- In the **logic view** we show that the proposal begins by redefining the basic elements of the architecture and by adding a new layer, the service layer, which is the solution offered by the Soa2m architecture to address the working hypothesis. Thus, we show that the proposal is based on the exploration of service-oriented interfaces in order to achieve the delegation of the user interface management.

First, we presented the basic elements of the architecture, the MMModule basic API's, and its taxonomy consisting on two kind of modules: the **Component** and the **Manager**, which is a helper sub-system to achieve the **Component** tasks. We emphasize in the addition of advertisement features by the wrapping of modules by entities named Services: the **Outgoing Services** that can advertise its behavior and interfaces and communicate externally, and the **Shy Services** that do not advertise its features.

Second, we present the functional layers of the Soa2m architecture, and the conceptual model behind these layers to finally show the structural relations and the building blocks proposed to handle the layer's functionalities. This descriptions allow us to highlight some irresoluted points raised by the status of «work in progress» of the MMI Framework & Architecture recommendation and to give some proposals on this direction. As a result, third, we explain the functional design patterns of the architecture, because multimodal systems may occur in a wide range of possibilities and the Soa2m architecture proposal is intended for supporting light as well as complex implementations in a designer-friendly approach.

- In the **process view** we presented how the MMModules communicate in run-time by introducing the communication model and its rules: the communication policy for Soa2m Components, Managers and Services. This descriptions allow us to highlight some undefined points about the communication policy and some legacy tendencies of the MMI Framework & Architecture and again, to give a proposal on this topic.
- In the **implementation view** we present the mechanism proposed for the creation of components, how these components can be deployed and the message model of Soa2m, which describes: how messages are transmitted, the structure of message and the relationship between senders and receivers. We explained how modules in Soa2m are created by factories, whit an HTTP client-server approach, and how the MMI Life-Cycle events are transmitted wrapped in web services structures, called envelopes and implemented as XML documents (SOAP) or javascript messages (JSON). Then, we described the communication protocol provided in the MMI Framework & Architecture to confront it to the Soa2m proposal: a mixed solution using push and pull techniques in order to transmit interaction commands to a multimodal user interface. To this extent, we introduced the proposition of an adaptive pull mechanism based on the use of a timeout triplet structure and two new events: the checkUpdate event and the UIUpdate notification.

- In the **physical view** we explain «the management of the remote user interface» in the Soa2m distributed deployment. This management is made according a delegation principle in which the client application delegates to the *multimodal system* the management of the composition of multimodal services in its front-end. Finally, the selection of the Soa2m architecture style and its previously detailed proposals respond to a series of functional and non-functional requirements.

Functional evaluation of the proposal

In this section we presented the propositions offered by the Soa2m approach to cover the six functional requirements derived from the study of the multimodal architectures in recent systems in section 1.3.

These architectural propositions are summarized in the [**Table 2.1.1**] as the addition of some extensions to an existent standard and the provision of some architectural solutions to allow these enhancements:

Functional Requirements	**RF1.** The *mode*, *modality* and *media* integration (*fusion*) and restitution (*fission*),	Definition of the Layer 2: Multimodal Participation
		Definition of the Layer 4: Multimodal Orchestration
	RF2. The bidirectional nature and the symmetry of the system,	Proposition of input>output pipelines.
		Proposition of a Interaction Manager as a meaning generator.
	RF3. An appropriate real-time sensing and responding at the sensorial, functional and semantic levels,	Definition of the Layer 5: State Handling
		Proposition of a Interaction Manager as a meaning generator.
	RF4. The management of the interaction cycles locally and globally,	Proposition of a State Manager as a context state provider.
		Proposition of an Advertise Manager as a registry of the available global services wrapping modalities and features.
	RF5. The support of incremental processing of incomplete and dynamic data,	Proposition of three data structures for the State Manager : situation, episode and outcome.
	RF6. The integration of embodied and situated knowledge about the interaction situation; covering devices, users, domain and environment and context of usage.	Definition of the Layer 6: Decision Handling
		Proposition of three data structures for the State Manager : situation, episode and outcome.
		Definition of four conceptual layers: sensory, functional, semantic and will.
		Proposition of an Advertise Manager to allow the semantic annotation of processes

Table 2.1.1: Architectural propositions to address the functional requirements.

Non-Functional evaluation of the proposal

Thus, the extensions offered by the Soa2m architecture are enhancements to an existent model. As a model, it is evaluated not only in terms of its functional requirements (that can evolve over time) but mostly in terms of non-functional requirements that ensures its capability to be adopted by a large community of developers.

As a result, the Soa2m architecture must also respond to these non-functional requirements. It was designed with this explicit goal, focusing on three perspectives: the non-functional requirements at design-time, the non-functional requirements at run-time and finally, the non-functional requirements affecting the user interaction with the system or the designer interaction with the implementation proposal.

[**Table 2.1.2**] presents the architectural propositions advanced by the Soa2m approach in regard to the non-functional requirements, at design-time:

Design-Time : Non-Functional Requirements	**RD1.** Extensibility	Proposition of a communication mechanism based on an adaptive pull technique for implementations willing to avoid persistent connections.
		Proposition of a Json data structure of messages for implementation willing to enhance network traffic performance.
		Proposition of factories for the creation of components to allow the addition of new configuration of components on the same architectural basis.
		The pursuing of the generic approach of a standardised architecture of reference.
		The proposition of a generalised use of recursion with complex modality components.
		The proposition of Managers as subsystems of Components, allowing the extension of its functionalities.
		The proposition of services as wrapper interfaces for modules.
		The proposition of advertised services
	RD2. Flexibility	Definition of the Layer 2: Multimodal Participation
		Proposition of factories for the creation of components to allow the addition of new configuration of components on the same architectural basis.
		The proposition of services as wrapper interfaces for modules that can change keeping the interface stable.

Design-Time : Non-Functional Requirements	**RD2.** Flexibility	The proposition of an access mechanism to data coming from the modeling processes.
		The proposition of advertised services
	RD3. Portability	The adoption of the Modality Component abstraction for input and outputs.
		The optional aspect of the service wrapping mechanism.
	RD4. Reusability	The common mechanism of interface and request in modules.
		The proposition of services as wrapper interfaces for modules that can be composed.
	RD5. Modifiability	Proposition of factories for the creation of components to allow the addition of new configuration of components on the same architectural basis.
		The common mechanism of interface and request in modules.
		The proposition of services as wrapper interfaces for modules.
	RD6. Maintainability	The proposition of a feedback service to monitor the multimodal session state and the management information.
	RD7. Completeness	Proposition of a communication mechanism based on an adaptive pull technique for implementations willing to avoid persistent connections.
		Proposition of a Json data structure of messages for implementation willing to enhance network traffic performance.
		Definition of the Layer 6: Decision Handling
		The proposition of a access mechanism to data coming from the modeling processes.
	RD8. Consistency	The pursuing of the generic approach of a standardized architecture of reference.
		The expressive naming convention used.
		The proposition of advertised services
		The adoption of the Modality Component abstraction for input and outputs.
		The use of the Interaction Life-Cycle Events as a basis of the communication for all modules and services.
		Proposition of a Controller Component encompassing the control features.
	RD9. Testability	*NOT ADDRESSED in the Architecture Proposal*
	RD10. Auditability	Proposition of an Advertise Manager as a registry of the available services.
		Proposition of a State Manager as a state provider.

Table 2.1.2: Architectural propositions to address non-functional requirements at design-time.

The table shows an issue that is not addressed in the architectural proposal: the testability. It will be treated in the implementation side and described in section 2.3 The Soa2m Tools implementation.

In [**Table 2.1.3**] we enumerate the architectural propositions of Soa2m approach of run-time non-functional requirements:

Run-Time : Non-Functional Requirements	**RR1.** Interoperability	The pursuing of the generic approach of a standardized architecture of reference designed for interoperability issues.
		Proposition of a Json data structure of messages for compatibility with existent web services implementations.
		Definition of four conceptual layers: sensory, functional, semantic and will.
		Definition of the Layer 7: Application Logic based on web services technologies.
		Definition of the Layer 2: Multimodal Participation.
	RR2. Performance	Proposition of a communication mechanism based on an adaptive pull technique for implementations willing to avoid persistent connections.
		Proposition of a Json data structure of messages for implementation willing to enhance network traffic performance.
	RR3. Reliability	Proposition of a communication mechanism based on an adaptive pull technique mixed with a push technique to shortcut the mechanism in cases of need.
	RR4. Availability	Proposition of three data structures for the State Manager : situation, episode and outcome.
		Proposition of a Json data structure of messages for implementation willing to enhance network traffic performance.
		Definition of four conceptual layers: sensory, functional, semantic and will.
		Proposition of an Advertise Manager as a registry of the available services.

Run-Time : Non-Functional Requirements	**RR5.** Scalability	The proposition of Managers as subsystems of Components, allowing the extension of its functionalities.
		Definition of the Layer 7: Application Logic based on web services technologies.
		Definition of the Layer 2: Multimodal Participation with the abstraction of Modality Components
	RR6. Security	The proposition of communication policies protecting the Data Component.
		The proposition of tokens in the payload of messages.
		The proposition of a registering mechanism handled by the State Manager.
	RR7. Manageability	*NOT ADDRESSED in the Architecture Proposal*
	RR8. Supportability	Definition of four conceptual layers: sensory, functional, semantic and will to support the modeling of the participants on the interaction.

Table 2.1.3: Architectural propositions to address non-functional requirements at run-time.

Finally [**Table 2.1.4**] list the architectural propositions advanced by the Soa2m approach in regard to the non-functional requirements for the interaction time.

Interaction-Time : Non-Functional Requirements	**RI1.** Accessibility	The proposition of a Knowledge Manager and Decision Manager to adapt the system to a high number of persons.
		Definition of the Layer 2: Multimodal Participation
		The proposition of semantic modeling processes for the management of users and interaction conditions.
	RI2. Usability	Definition of the Layer 2: Multimodal Participation with the generic abstraction of the Modality Components.
		The proposition of patterns and naming conventions for the implementations possibles of the architecture.
	RI3. Configurability	The adoption of the Modality Component abstraction for input and outputs.
		The use of factories for the creation of Components and Managers
		The proposition of semantic modeling processes for the management of users and interaction conditions.
	RI4. Buildability	The proposition of a generalized use of recursion with complex modality components.
		The use of factories for the creation of Components and Managers
		The proposition of patterns and naming conventions for the implementations possibles of the architecture.
	RI5. Automation	Proposition of a SOAP data structure of messages for compatibility with existent web services automatic composition techniques.
		The proposition of services as wrapper interfaces for modules that can change keeping the interface stable.
	RI6. Integrability	Proposition of a communication mechanism based on an adaptive pull technique for implementations willing to avoid persistent connections.
		Proposition of a Json data structure of messages for compatibility with existent web services implementations.
		The pursuing of the generic approach of a standardized architecture of reference.
		Definition of the Layer 7: Application Logic based on web services technologies.
		The proposition of a Shy service contained in Outgoing Services without the need of a description or advertisement of Shy services.
		The proposition of a delegation mechanism for remote interfaces.

Table 2.1.4: Architectural propositions to address the non-functional requirements at interaction-time.

In conclusion, this section presented the Soa2m proposal, and its main architectural concerns, oriented to collaborate with the MMI Framework & Architecture recommendation in aspects like the extensibility, consistency, integrability, flexibility and interoperability. The following section will complete this description with the presentation of the semantic approach of the Soa2m services-oriented architecture.

2.2 A Semantic SOA Architecture.

In the last section we presented the Soa2m architecture from a service-oriented perspective and we indicate as a contribution the proposal of an abstract **layer 7:** Application Logic[108] and its Advertise Manager responsible of the management of services.

The goal of the current section is to present this complementary part of the approach, namely, the enrichment of services with metadata annotations coming from the semantic web[109] services community. We expect that this enrichment will enhance the management of services of a *multimodal system* created according the Soa2m architecture proposal.

Until now, we have considered a Multimodal Service as a set of functionalities associated with a process or a system that performs a task in a MMModule (Component or Manager). For our purposes, a Multimodal Service is any wrapped functionality of a Modality Component which uses one or multiple devices.

Characteristics of Modality Component Services

Therefore, these services can handle input or output interaction in one or more devices, *sensors*, *effectors*, players (e.g. for virtual reality media display), on-demand distributed[3]recognizers (e.g. for natural language recognition) or user interface widgets (e.g. maps for geo-location display, MPEG-U TV guides). These services can also be composed in real-time using the *fission* and *fusion* mechanisms and they are the basis of the system's behavior towards the user.

With the *fusion* mechanism, for example, we achieve the integration of data coming from a set of input Services representing the Modality Component functionalities. This integration produce specific and unified information about an interaction cycle executed in multiple *modes*.

On the other hand, with the *fission* mechanism we compose a single message on some combination of the available Services representing Modality Components for more than one sensorial *mode*. This process occurs before the processes dedicated to the information rendering or to the media instantiation. The goal is to generate an adequate message, according to some metadata. This metadata can describe the application features, the space of use, the current activity or the user profile.

As multimodal interaction in large-networks takes place in highly dynamic and unpredictable conditions, reactive and proactive planning are needed in both mechanisms in order to respond to any incoming interaction. This process requires making decisions that maximize the utility of the available components. Based on these choices, it is possible to set goals and achieve them during a defined period of time. Therefore, intelligent decision making is a key pillar for action planning in interaction management or for any kind of coordination of inputs / outputs.

The system needs to "understand" internal and external states, time, goals, entities (devices, services, *modalities*, *media*), participants (users or systems), properties, categories and relations between them. As we show in section 1.2. Definition of a Multimodal System, decision techniques are already available to support these choices, but semantic annotation of entities and process are still needed.

Outline

The following section will present the Soa2m semantic approach concerning the metadata needed for multimodal processes and data in multimodal systems. This will be made from three perspectives: the requirements perspective, the annotation techniques perspective and finally the data model perspective.

First, we will present the requirements for the management of multimodal services, regarding the advertisement of services, their discovery and the registration of services. This will let us to detail some functional responsibilities of the Advertise Manager in the Soa2m architecture, and also it will give us a context for the description of multimodal processes and data.

108 This layer was defined as the logical application of the level four of the conceptual model of the Soa2m multimodal approach, namely the Will level, covering the explicit intent of the actions of the entities.

109 The semantic web is about adding machine-processable semantics to data. The computer can "understand" the information and therefore process it on behalf of the human user. Web services try to employ the web as a global infrastructure for distributed computation, integration and automatization of processes: it is the place where global computing will be realized.

Thus, in a second moment, we will explain the description of services in Soa2m as processes representing a consummation contract or an intent. The description will be oriented to the discovery and registering mechanism[110], given that the main contribution of this thesis is around this issue, namely: how to handle the dynamic availability of Modality Component Services through the Advertise and State Managers. In this section we will only present the annotation mechanism proposed using the existing web semantic technologies, letting for the final part the explanation about the data model used in this annotation process.

Third, we will describe how multimodal entities are modeled in Soa2m in order to support the decision making mechanism for the context-awareness of the system. In this section we will introduce the Soa2m ontologies and the proposal of a multimodal situation data model.

2.2.1 The Requirements for Multimodal Services

As we already discussed, a *multimodal system* must discover which services representing the Modality Components are available at any moment in a target location or network. This requirement is called the «discovery of services».

Furthermore, these resources must be allocated to more complex multimodal services and dynamically composed to provide a more complete service in a particular situation and with a particular form.

This requirement is called the «querying of services».

The system must also identify the needs of the application as the expected final goal of the multimodal service and confront them to the service offer advertised in some description.

In this way, the system can decide what is the best available form to compose the service in this particular user's context and what is the acceptable minimum of quality of service in order to achieve the fixed goal. This requirement is called the «advertisement (or publishing) of services».

Finally, the system must have an adequate and generic mechanism of requesting that can be stored and used in a large number of multimodal consumption cases in similar spaces or networks. This requirement is called the «registering of services».

Given the diversity of devices and networks that are used in multimodal systems, this structure must be as open as possible to many user-preferred operating systems, devices capabilities and networks when described with from a standardization point of view.

For us, the MMI Architecture used in conjunction with semantic services tools can address these three challenges for the management of Modality Components.

In this section we will describe each of these requirements under the light of the needs for a standardized multimodal service-oriented architecture.

2.2.1.1 Advertisement

Advertisement of MC Services[111] is one of the most influential criteria to evaluate how well the system can be adapted to the context and environment situation.

Definition of the advertisement mechanism

Advertisement starts with the description mechanism of the existing services. This description can be provided by the Soa2m architecture, by MC Services furnisher or by the client application at design-time.This allows the *multimodal system* to reach correctness in the MC Services retrieval, because a pertinent and expressive description enables the result to match more closely to the application's or user's queries.

110 This means that our examples will illustrate mostly the discovery processes, which are previous to the interaction cycle *fusion* or *fission*.

111 To facilitate reading comprehension, from now on we will call the Modality Component Services : «MC Services».

On the other hand, Advertisement also affects the completeness (**RD7**) in the MC Services retrieval. To return all matching instances registered and corresponding to the query, the query criteria must match to some basic attributes defined in the MC Service Description.

Types of Description

A service description that lists some multimodal attributes of the wrapped Modality Component and the operations offered is then, required. This description of the MC Service features can be expressed in multimodal systems as a data structure (e.g. YAML [EVANS-YAML 2001]), as a simple attribute-value pair list, as a list of attributes with hierarchical tree relationships, as a WSDL document [W3C-WSDL 2007] or as a manifest document.

In Soa2M the description of the MC Services is suggested in three of these forms[112]: a data structure (a YAML file), a manifest document (a RFD file [W3C-RDF 2004]) or a WSDL document. All of these formats can be eventually enriched with semantic web technologies but also they can be used as they are.

Proposal: Advertisement with YAML files

The first type of description is intended to reach a basic list of attributes and operations in a human-readable data serialization format: a YAML file. For example, attributes like the MC Service life (which allows to handle its availability) and the type of Modality Component wrapped by the service (which allows to infer some generic features), can be expressed as:

```
Life: {
  GoodFrom : 2012-04-21
  GoodUntil : 2012-04-21
  NotGoodAfter : 2015-04-21
  }
Type: |
  Relational : OUTGOING
  Temporal : CHRONICLER
  Decisional : SMART
   }
```

Proposal:Advertisement with RDF ontologies

The second format is intended to describe the attributes of MC Services linking them with the semantic web[6] to avoid ambiguity in the description. Then, the same attributes of the example above can be expressed as :

```
< Life >
    <rdf:Seq>
        <rdf:li><rdf:Description>
                <xm:GoodFrom rdf:datatype="&xsd;dateTime"> 2012-04-21T03:33:33 </xm:GoodFrom >
        </rdf:Description></rdf:li>
        <rdf:li><rdf:Description>
                <xm:GoodUntil rdf:datatype="&xsd;dateTime"> 2013-09-21T03:33:33 </xm:GoodUntil >
        </rdf:Description></rdf:li>
        <rdf:li><rdf:Description>
                <xm:NotGoodAfter rdf:datatype="&xsd;dateTime"> 2013-09-21T03:33:33 </xm:NotGoodAfter >
        </rdf:Description></rdf:li>
    </rdf:Seq>
</ Life >
< Type >
    <rdf:Seq>
        <rdf:li><rdf:Description>
                <xm:Relational rdf:datatype="&xsd;string"> OUTGOING </xm:Relational  >
        </rdf:Description></rdf:li>
        <rdf:li><rdf:Description>
                <xm:Temporal rdf:datatype="&xsd;string"> CRONICLER </xm:Temporal >
        </rdf:Description></rdf:li>
        <rdf:li><rdf:Description>
                <xm:Decisional rdf:datatype="&xsd;string"> SMART </xm:Decisional >
        </rdf:Description></rdf:li>
    </rdf:Seq>
</ Type >
```

112 This section, oriented to explain our requirements will be completed with the detailed description of the annotation technologies and the annotation proposal in Soa2m in section 2.2.2. The Semantic Annotation of Services.

Thus, as the example illustrates, this second format add some important information to enhance the parsing process. It includes for example, dataTypes for each attribute or the fact that the attributes are ordered triples with the use of the <rfd:Seq> tag (line 2 and 15).

Proposal: Advertisement with WSDL files extended with OWL

Finally the third format is intended to describe the processes provided by tha Modality Component Service and to enhance the contextual-awareness. This is realized by the use of semantic descriptions in service inputs, outputs, preconditions, effects and situations of usage [6]. If we suppose that the MC Service described above is an Availability Listener (a MC Service responsible to monitor changes on the availability in the components used for the *fusion* and *fission* processes) , the WSDL description in Soa2m will be linked (line 3) with a semantic annotation file:

```
1     <wsdl:interface name="ListenerInterface" >
2             <wsdl:operation name="UPnPAvailabilityList" pattern="http://www.w3.org/ns/wsdl/in-out"
3                 sawsdl:modelReference="ser:Listener#UPnPAvailabilityList" >
4
5                      <wsdl:input messageLabel="UPnPAvailabilityListInput"
6                          element="tns:UPnPAvailabilityListRequest" />
7
8                      <wsdl:output messageLabel="UPnPAvailabilityListOutput"
9                          element="tns:UPnPAvailabilityListResponse" />
10             </wsdl:operation>
11     </wsdl:interface>
```

This file will describe the attributes already treated above, Life and Type, and some important relational constraints between them, like heritage (line 18) and cardinality (line 20). In the RDF context the heritage property (the <rdfs:subClassOf> tag) informs that the entity MUST, by heritage, contain these data while the cardinality property (the <owl:cardinality> tag) informs that the attributes are unique and mandatory.

```
1  <?xml version='1.0' ?>
2  <!DOCTYPE uridef[
3  <!ENTITY rdf     "http://www.w3.org/1999/02/22-rdf-syntax-ns">
4  <!ENTITY rdfs    "http://www.w3.org/2000/01/rdf-schema">
5  <!ENTITY xsd     "http://www.w3.org/2001/XMLSchema">
6  <!ENTITY owl     "http://www.w3.org/2002/07/owl">
7  <!ENTITY soa2m   "http://localhost/soa2m/ont/soa2m.owl">
8  ]>
9
10  <rdf:RDF
11   xmlns:rdf= "&rdf;#"
12   xmlns:rdfs= "&rdfs;#"
13   xmlns:xsd = "&xsd;#"
14   xmlns:owl = "&owl;#"
15   xmlns:soa2m = "&soa2m;#" >
16
17      <owl:Class rdf:ID="MC">
18          <rdfs:subClassOf> <owl:Restriction>
19              <owl:onProperty rdf:resource="#hasLife" />
20              <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
21          </owl:Restriction> </rdfs:subClassOf>
22          <rdfs:subClassOf> <owl:Restriction>
23              <owl:onProperty rdf:resource="#hasType" />
24              <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
25          </owl:Restriction> </rdfs:subClassOf>
26      </owl:Class>
27
28      <owl:Class rdf:ID="Listener">
29          <rdfs:subClassOf rdf:resource="#MC" />
30      </owl:Class>
31
32      <owl:Class rdf:ID="AvailabilityListener">
33          <rdfs:subClassOf rdf:resource="#Listener" />
34      </owl:Class>
35
36      <owl:Class rdf:ID="UPnPAvailability">
37          <rdfs:subClassOf rdf:resource="#AvailabilityListener" />
38      </owl:Class>
39
40      <owl:Class rdf:ID="UPnPAvailabilityList">
41          <rdfs:subClassOf f rdf:resource="#UPnPAvailability" />
42      </owl:Class>
```

To these constrains the description adds their relationships with the MC Service definition (lines 28>42) and finally, the annotations of the attributes of the service (lines 44>73, 74>116, 126>136) :

```
44    <owl:Class rdf:ID="Life">
45        <rdfs:subClassOf> <owl:Restriction>
46            <owl:onProperty rdf:resource="#hasStandBy" />
47            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
48        </owl:Restriction> </rdfs:subClassOf>
49        <rdfs:subClassOf> <owl:Restriction>
50            <owl:onProperty rdf:resource="#hasUnavailability" />
51            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
52        </owl:Restriction> </rdfs:subClassOf>
53        <rdfs:subClassOf> <owl:Restriction>
54            <owl:onProperty rdf:resource="#hasDead" />
55            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
56        </owl:Restriction> </rdfs:subClassOf>
57    </owl:Class>
58
59    <owl: DatatypeProperty rdf:ID="hasStandBy">
60        <rdfs:domain rdf:resource="#GoodFrom"/>
61        <rdfs:range rdf:resource="&xsd;dateTime"/>
62    </owl:DatatypeProperty>
63
64    <owl: DatatypeProperty rdf:ID="hasUnavailability">
65        <rdfs:domain rdf:resource="#GoodFrom"/>
66        <rdfs:range rdf:resource="&xsd;dateTime"/>
67    </owl:DatatypeProperty>
68
69    <owl: DatatypeProperty rdf:ID="hasDead">
70        <rdfs:domain rdf:resource="#GoodFrom"/>
71        <rdfs:range rdf:resource="&xsd;dateTime"/>
72    </owl:DatatypeProperty>
73
74    <owl:Class rdf:ID="Type">
75        <rdfs:subClassOf> <owl:Restriction>
76            <owl:onProperty rdf:resource="#hasDescription" />
77            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
78        </owl:Restriction> </rdfs:subClassOf>
79        <rdfs:subClassOf> <owl:Restriction>
80            <owl:onProperty rdf:resource="#hasHistory" />
81            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
82        </owl:Restriction> </rdfs:subClassOf>
83        <rdfs:subClassOf> <owl:Restriction>
84            <owl:onProperty rdf:resource="#hasWill" />
85            <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger"> 1 </owl:cardinality>
86        </owl:Restriction> </rdfs:subClassOf>
87    </owl:Class>
88
89    <owl: DatatypeProperty rdf:ID="hasDescription">
90        <rdfs: domain rdf:resource="#Relational"/>
91        <rdfs: range rdf:resource="&xsd;string"/>
92    </owl:DatatypeProperty>
93
94    <owl: DatatypeProperty rdf:ID="hasHistory">
95        <rdfs: domain rdf:resource="#Temporal"/>
96        <rdfs:range rdf:resource="&xsd;string"/>
97    </owl:DatatypeProperty>
98
99    <owl: DatatypeProperty rdf:ID="hasWill">
100      <rdfs: domain rdf:resource="#Decisional"/>
101      <rdfs: range rdf:resource="&xsd;string"/>
102   </owl:DatatypeProperty>
103
104   <owl:ObjectProperty rdf:ID="hasLife">
105      <rdfs:comment> A Modality Component have at most one life. </rdfs:comment>
106      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
107      <rdfs:domain rdf:resource="#MC"/>
108       <rdfs:range rdf:resource="#Life"/>
109   </owl:ObjectProperty>
110
```

```
  <owl:ObjectProperty rdf:ID="hasType">
     <rdfs:comment> A Modality Component have at most one Type. </rdfs:comment>
     <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
     <rdfs:domain rdf:resource="#MC"/>
     <rdfs:range rdf:resource="#Type"/>
  </owl:ObjectProperty>

  <UPnPAvailabilityList rdf:ID="MC_1234">
       <hasLife rdf:resource="#Life" />
 </UPnPAvailabilityList>

 <UPnPAvailabilityList rdf:ID="MC_1234">
       <hasType rdf:resource="#Type" />
 </UPnPAvailabilityList>

 <Life rdf:ID="Life">
       <GoodFrom> 2012-04-21T03:33:33 </GoodFrom>
       <GoodUntil> 2013-09-21T03:33:33 </GoodUntil>
       <NotGoodAfter> 2013-09-21T03:33:33 </NotGoodAfter>
 </Life>

 <Type rdf:ID="Type">
       <Relational> OUTGOING </Relational>
      <Temporal> CRONICLER </Temporal>
       <Decisional> SMART </Decisional>
 </Type>

</rdf:RDF>
```

Semantic technologies provides more expressivity in the description at the price of higher work load

As we can see in this example, the three kinds of description increase in expressivity but in return, they also increase in difficulty and verbosity. Semantic technologies enriches the description of attributes but also demand a higher level of complexity and a more important information and work overload. For this reason, in Soa2m we propose these three degrees of expressivity in the description formats, for example, by proposing a more accessible format for users with simpler needs.

The description of the MC Services in Soa2m cover also four categories of data. These data are more or less detailed depending on the implementation needs:

- Information to identify the MC Service : For example, an unique identifier of the wrapped Modality Component, its name, its type, the *mode* and *modality* supported, network address, port number, its container Device, constructor, version, transport protocol or communication default timeout (sleep time, life time and communication interval).

- Information about the behavior of the MC Service : The list of *media* handled by the MC Service; a list of the commands or actions to which the MC Service responds; the parameters for each action and the information variables that models the state of the MC Service. For example, information showing if the component processes continuous or discrete information.

- Information about the semantics of the MC Service for a specific domain: The MC Service's federation group (e.g. the Zone in AppleTalk Name Binding Protocol [Apple-AppleTalk 1985] or the DNS subdomain), its scope (e.g. like UA scopes in SLP [IETF-RFC2608 1999]), its category, its intent (e.g. like Implicit Intents in Android [Google-AIntent 2012] or Web Intents [W3C-WEBINTENTS 2012]) or any other semantic metadata.

- Information about the intention of the MC Service: This may cover the name of the organization providing the MC Service, the service level agreement, its authorizations, authentication procedures, privacy policies, its business type, its access point. And finally, complementary information about invocation policies or implementation details that can be application-specific or can follow some specification (e.g. UPnP [UPnP-Arch 2008], ECHONET [ECHONET-DeviceObjects 2002] or IANA [ICANN-IANA 1998]).

A data model with some examples of attributes specific to the multimodal domain are suggested in the Soa2m architecture to enhance interoperability, expressiveness and relevance in the description's content from a multimodal interaction point of view.[113]

In this way, this kind of data model can be used also as a support for the annotation, the modality selection and the *fusion* and *fission* mechanisms in analyzers or synthesizers implemented in the *multimodal system*.

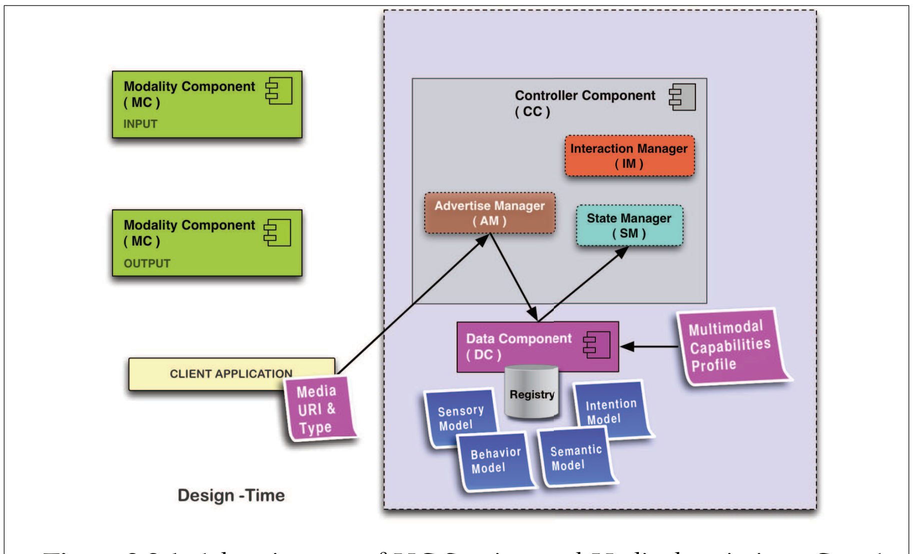


Figure 2.2.1: Advertisement of MC Services and Media description - Case 1

Types of advertisement in Soa2m

For example, [**Figure 2.2.1**] shows a description of multimodal capabilities in a manifest provided at design time by the Soa2m architecture (on its Data Component). This profile can be completed by an annotated list of *media* (e.g. using EMMA [W3C-EMMA 2009]) and its URI's, declared at design time by the client application and to be used by the Modality Component. In this case, the information is previously declared to the multimodal system before the run-time, in the phase of configuration of the *multimodal system* services.

Another case at design time can be [**Figure 2.2.2**] where the description of multimodal capabilities in a manifest, and the annotated list of *media*, are given by the Soa2m architecture in the Modality Components, in some embedded file or advertisement service, provided by a nested controller, for example.

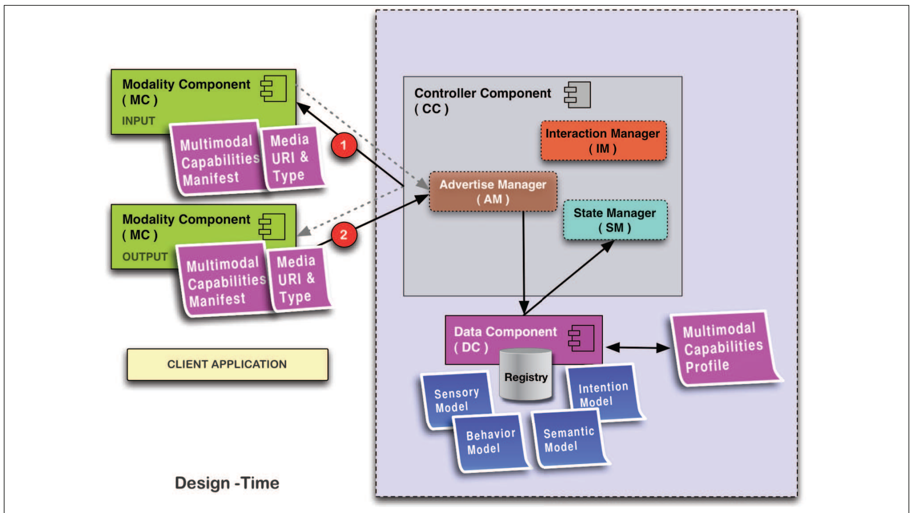

Modality Component (MC) INPUT
Multimodal Capabilities Manifest
Media URI & Type
Modality Component (MC) OUTPUT
Multimodal Capabilities Manifest
Media URI & Type
1
2
CLIENT APPLICATION
Design -Time
Controller Component (CC)
Interaction Manager (IM)
Advertise Manager (AM)
State Manager (SM)
Data Component (DC)
Registry
Multimodal Capabilities Profile
Sensory Model
Intention Model
Behavior Model
Semantic Model

Figure 2.2.2: Advertisement of MC Services and Media description - Case 2

113 This data model will be presented in infra section 2.2.3 Semantic Description of Multimodal Data

In this case, the advertisement can be demanded by the Controller Component (1) or declared by the Modality Component (2) to a dedicated service in the Advertise Manager. [**Figure 2.2.2**] Finally this information is processed according to the data models, to store it in a profile and to update the current state of the system.

A third case at design time is when the description of multimodal capabilities manifest, and the annotated list of *media,* of all the MC Services to be used are provided by the client application to the Soa2m architecture, which will provide mostly the *fusion* and *fission* support. [**Figure 2.2.3**] Again, the information is processed according to the data models, to save it in a profile structure and to update the state.

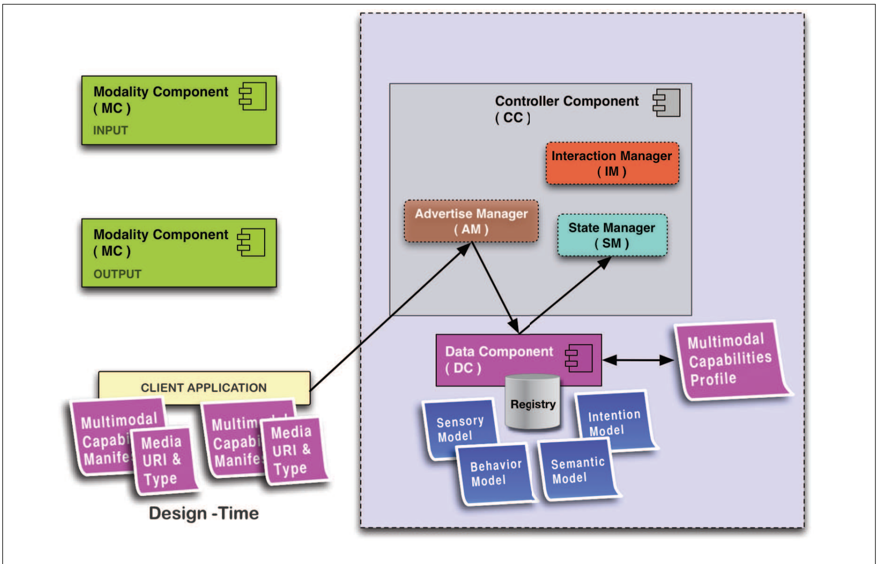


Figure 2.2.3: Advertisement of MC Services and Media description - Case 3

Finally, at run-time [**Figure 2.2.4**], when a multimodal session is already active, a user present and an interaction cycle is started; the Controller Component can «scan» [114] the environment to find new Modality Components and recover its description and *media* list (1). Otherwise, a new Modality Component Service can advertise its description of capabilities and *media* (2) to a dedicated service in the Advertise Manager. In this case, the information must be processed according to the data models, to store the profile structure and to dynamically update the state of the *multimodal system.*

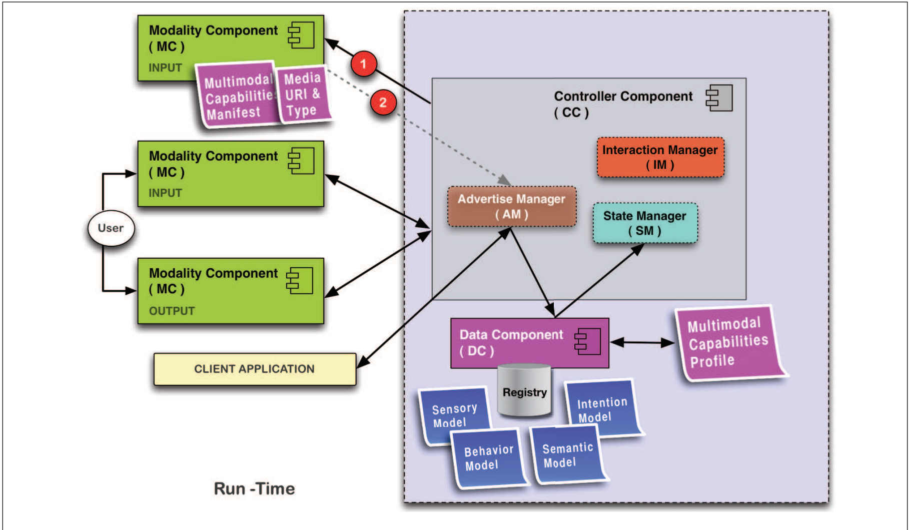


Figure 2.2.4: Advertisement of MC Services and Media description - Case 4

114 See infra section 2.2.2.1 Discovery.

To sum up, in Soa2m the MC Services advertisement is based on three types of descriptions enriched with semantic web technologies. The data described by these three description formats are structured in four categories: identity, behavioral, semantic and intentional data.

Conclusion

The descriptions can be advertised at design time or run-time. The advertisement can be initiated by the Controller Component or declared by the Modality Component to the Advertise Manager using a Soa2m service provided for this task.

2.2.1.2 Discovery

Service discovery allows the automatic detection of non-advertised MC Services at load and run-time. It also allows to update the profiles of advertised services already stored in a registry.

In Soa2m the discovery process in divided on four phases:

Discovery Steps

- The **intention** discovery concerns the conceptual level related to the interpretation and motivation for the service[115]. This is the abstraction of the provider and user intentions and their commitments to a pre-defined, reusable, and formalized structure; to be understood and used by humans and/or computers.
- The **semantic** discovery concerns the conceptual level related to the embodied, situated and social knowledge about these goals and commitments in a specific domain [116]. This is the abstraction to be understood and used by humans and/or computers of the situational knowledge related to the goal and commitments structures.
- The **behavior** discovery deals with the matching of formalized intentions and commitments, with formalized and well described processes and outcomes to be used mostly by computers.
- The last phase, the **capacities** discovery, uses the formalized behavior matches to access the real processes behind the MC Services behavior interfaces. Then, finally it performs a detailed check on what service fulfill the requester goal, how it fulfill it and with which kind of modes, modality and media.

Depending on the implementation, only some of these phases can be executed.

In most of the cases, only the two last phases are implemented, following a system-oriented and a functional perspective. Nevertheless, in systems with the need of context-awareness or intelligent user-oriented behavior, the first two phases can be also supported.

In Soa2m, the annotation of services for **intention discovery** and **semantic discovery** is made using the third format listed in section 2.2.1.1 Advertisement. This covers the use of semantic web technologies in the service description made with WSDL .

WSDL is an XML-based language that is used to describe the functionality offered by a Web service. The building block of SOA-based solutions is the self-describing Web service that can be reused across various applications. WSDL was created specifically for this purpose and describes the data elements, operations and message bindings. However, WSDL descriptions are not sufficient to unambiguously decipher each operation process and requirements. The Semantic Annotations for WSDL (SAWSDL) recommendation [W3C-SAWSDL 2007] overcomes this limitation by adding meta-data to the WSDL elements using the *modelReference* attribute that contains a reference to a concept in the ontology defining the semantics of the annotated element.

Extension of WSDL with SAWSDL references to a model

With this goal, we use the *modelReference* attribute, which relies the description with two types of ontologies that will provide the knowledge about the user intent and the system goal, and the semantics of the overall situation (line 1).

115 See supra section 2.1.1.2 Abstract Layers of the Architecture.

116 See supra section 2.1.1.2 Abstract Layers of the Architecture.

These are the **usage** ontologies in Soa2m represented by the prefix « **use** ». For example, in line 1 the service description is linked to an ontology describing a MC Service intended to «Listen». An usage ontology will then describe what the «Listening» situation means in the context of the MC services discovery, and how this task of listening can be interpreted as monitoring a service availability.

```
1     <wsdl:service name="MCs_GpacServicesListener"
       interface="tns:ServicesListenerInterface"  sawsdl:modelReference="&use;continuous/Listener.owl#myServicesListener" >
2           <wsdl:endpoint name="ServicesListenerEndpoint"
             binding="tns: ServicesListenerSOAPBinding"  address="tns:MC_5678/MCs_GpacServicesListener.php" />
3     </wsdl:service>
```

Proposal: An ontology to describe the generic situation of usage

In other words, the linked ontology, will describe the generic situation of usage of this kind of MC service from a high level perspective, its common sense requirements and limitations and its common sense definition (what availability is, what is to «listen» what is needed to listen). It will describe in which minimal conditions a service of this kind must act, who is involved in this service what is its generic intention and functioning.

In contrast, the **service** ontologies in Soa2m are represented by the prefix « **ser** » and they are dedicated to give a complete and semantically rich description for the concrete **behavior discovery**.

The **service** ontology describe the IOPE properties[117] of the service using semantic services technologies proposed by the W3C[118].

For example, in line 3, the operation description is linked to a service ontology « **&ser;discrete/Listener#UPnPAvailabilityList** » giving more details about the behavior of the annotated operation: its inputs and outputs but also its preconditions and effects.

```
1  <wsdl:interface name="ListenerInterface" >
2    <wsdl:operation name="UPnPAvailabilityList" pattern="http://www.w3.org/ns/wsdl/in-out"
3                    sawsdl:modelReference="&ser;discrete/Listener#UPnPAvailabilityList" >

4      <wsdl:input messageLabel="UPnPAvailabilityListInput" element="tns:UPnPAvailabilityListRequest" />
5      <wsdl:output messageLabel="UPnPAvailabilityListOutput"  element="tns:UPnPAvailabilityListResponse" />
6    </wsdl:operation>
7  </wsdl:interface>
```

Proposal: An ontology to describe the processes of the service

In consequence, the role of a service « ser » ontology is to complete a simpler description of the service, avoiding redundancy by the use of complementary rich information about aspects of the service which can not be described using other means.

These simpler descriptions are used for the capacities discovery phase, and they can be expressed in the WSDL format (like in our example), but also with the RDF or the YAML format12, depending on the needs of the multimodal system.

They are restricted to describe the attributes (inputs, outputs, restrictions or communication modes) of the process and its datatypes. For example, a description for **capacities discovery** can be composed by the WSDL description of the operation given in the example code above in lines 2 to 6, and its corresponding datatype description in the lines 21 to 74 of the following code:

117 Atomic, Simple and Composite processes all have Inputs, Outputs, Preconditions and Effects : the IOPEs properties. In the case of an atomic process, the role of these properties is obvious. Every input to the service is designated as an input property of the atomic process, every conditional output, is designated as a conditional output property, and so on. In multimodal systems also processes can de described as symmetric cycles in which the outputs can be affected by the inputs to produce some effect according with some preconditions.

118 For more details about the implementation and a brief introduction to the semantic web technologies used, see infra section 2.2.2 Semantic Annotation of Services

```
21 <wsdl:types>
22    <xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
23     targetNamespace="http://localhost/soa2m/desc/wsdl/" >
24        <xs:element name="UPnPAvailabilityListRequest"
25         sawsdl:modelReference="&ser;discrete/Requester#myParametrizedRequest"
26         type="tns:checkUpdateRequestCT">
27           <xs:complexType name="checkUpdateRequestCT">
28               <xs:sequence>
29                   <xs:element name="context" type="mmi:context.optional.attrib"/>
30                   <xs:element name="source" type="mmi:source.attrib"/>
31                   <xs:element name="target" type="mmi:target.attrib"/>
32                   <xs:element name="type" type="mmi:type.attrib"/>
33                   <xs:element name="requestID" type="mmi:requestID.attrib"/>
34                   <xs:element name="updateType" type="soa2m:updateType.attrib" />
35                   <xs:element name="state" type="soa2m:state.attrib"/>
36                   <xs:element name="data" minOccurs="0" type="mmi:anyComplexType">
37                       <xs:complexType name="UPnPAvailabilityListParameters"
                          sawsdl:modelReference="&use;discrete/Listener#myAttentiveListener">
38                          <xs:sequence>
39                              <xs:element name="onlyTotal" sawsdl:modelReference="&use;discrete/Focus#myQuantity" />
40                              <xs:element name="serviceType" sawsdl:modelReference="&use;discrete/Focus#myType"/>
41                              <xs:element name="serviceHostName"sawsdl:modelReference="&use;discrete/Focus#myLocation" />
42                          </xs:sequence>
43                       <xs:complexType
44                   </xs:element>
45               </xs:sequence>
46           </xs:complexType>
47        </xs:element>

48        <xs:element name="UPnPAvailabilityListResponse"
49         sawsdl:modelReference="&ser;discrete/Responder#myFilteredResponse"
50         type="tns:checkUpdateResponseCT">
51               <xs:complexType name="checkUpdateResponseCT">
52                 <xs:sequence>
53                   <xs:element name="context" type="mmi:context.optional.attrib"/>
54                   <xs:element name="source" type="mmi:source.attrib"/>
55                   <xs:element name="target" type="mmi:target.attrib"/>
56                   <xs:element name="type" type="mmi:type.attrib"/>
57                   <xs:element name="requestID" type="mmi:requestID.attrib"/>
58                     <xs:element name="timeout" type="soa2m:timeout.attrib"/>
59                     <xs:element name="automaticUpdate" type="soa2m:automaticUpdate.attrib"/>
60                     <xs:element name="updateType" type="soa2m:updateType.attrib"/>
61                     <xs:element name="state" type="soa2m:state.attrib"/>
62                     <xs:element name="data" minOccurs="0" type="mmi:anyComplexType">
63                        <xs:complexType name="UPnPAvailabilityListResult">
64                          <xs:sequence>
65                              <xs:element name="total" sawsdl:modelReference="&use;discrete/Counter#myAddition" />
66                       <xs:element name="servicesList" sawsdl:modelReference="&use;discrete/Services#myUPnPServices"/>
67                          </xs:sequence>
68                        <xs:complexType
69                     </xs:element>
70                 </xs:sequence>
71               </xs:complexType>
72        </xs:element>
73    </xs:schema>
74 </wsdl:types>
```

In this way, the functional properties and datatypes are described to ensure the fourth phase of discovery. This phase represents the lower level of complexity in the description and the behavior abstraction. In our example, it shows that each parameter is part of the MMI recommendation and correspond to a precise dataType (lines 29 to 36, and 52 to 62), but also that the Request and the Response are completed by a service ontology, that will explain in generic terms the semantics and restrictions of this type of request (a parametrized request -line 25) and the response that we can expect (a filtered response-line 49).

Proposal: Four mechanisms of discovery

On the other hand, in Soa2m four mechanisms are used to discover MC Services in any of the discovery phases: fixed discovery, mediated discovery, active discovery or passive discovery.

-In **fixed discovery**, the Controller Component and the MC Services are assumed to know their address and the port number to listen. This information is previously declared at design-time and at load or run time the discovery becomes a confirmation of availability and correctness of the already known information. For example, in the case of the capacities discovery at load-time. [**Figure 2.2.5**], on bootstrapping, when the registry is not preconfigured and the cold start problem is present[119]. In this case, the Controller Component can send a push request via the MMI Status Event pair to the Modality Component asking for availability, description address (in the form of a manifest) and a *media* list, as showed in [**Figure 2.2.5**]. When it receives the description data, it gives this data to the Data Component. After processing according with the models, the Data Component finally creates the starting profile.

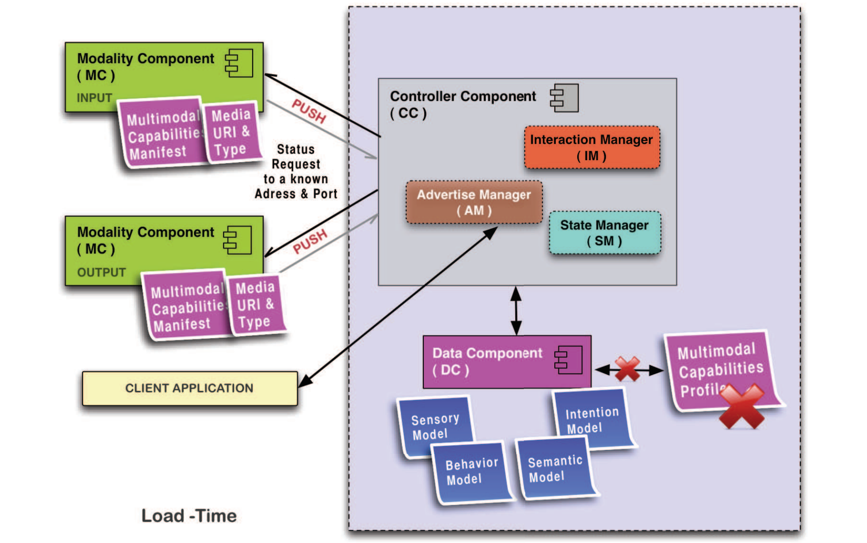

Modality Component (MC) INPUT
Multimodal Capabilities Manifest
Media URI & Type
PUSH
Status Request to a known Adress & Port
Modality Component (MC) OUTPUT
Multimodal Capabilities Manifest
Media URI & Type
PUSH
CLIENT APPLICATION
Controller Component (CC)
Interaction Manager (IM)
Advertise Manager (AM)
State Manager (SM)
Data Component (DC)
Multimodal Capabilities Profile
Sensory Model
Intention Model
Behavior Model
Semantic Model
Load -Time

Figure 2.2.5: Fixed Discovery of MC Services: Push Request

In other implementations, the MC Service can also announce its availability and description directly to a service provided by the Advertise Manager using the same MMI Event with a pull request at load-time. [**Figure 2.2.6**]

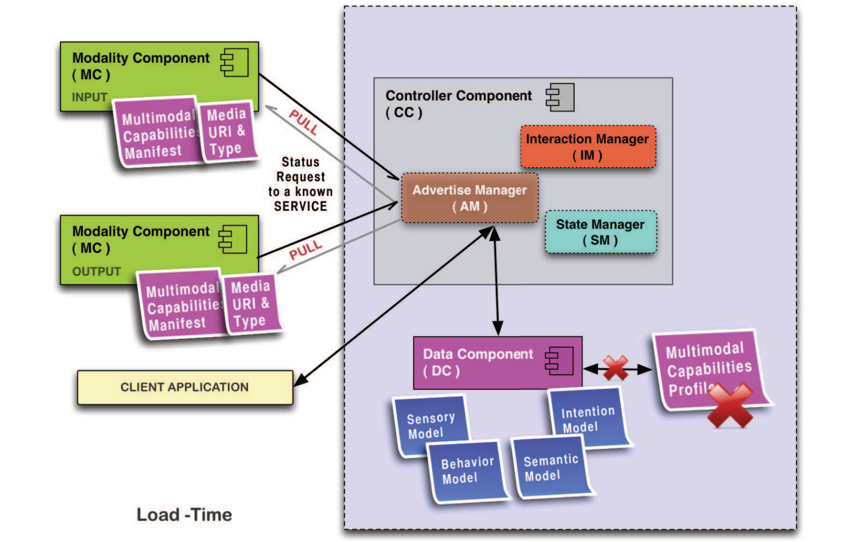

Modality Component (MC) INPUT
Multimodal Capabilities Manifest
Media URI & Type
PULL
Status Request to a known SERVICE
Modality Component (MC) OUTPUT
Multimodal Capabilities Manifest
Media URI & Type
PULL
CLIENT APPLICATION
Controller Component (CC)
Interaction Manager (IM)
Advertise Manager (AM)
State Manager (SM)
Data Component (DC)
Multimodal Capabilities Profile
Sensory Model
Intention Model
Behavior Model
Semantic Model
Load -Time

Figure 2.2.6: Fixed Discovery of MC Services: Pull Request

119 The cold start problem in multimodal systems involve the dynamic use of Modality Components: the system cannot use any MC Service about which it has not yet gathered sufficient information.

- In **mediated discovery**, the Controller Component can use the scanning features provided by the underlying network (e.g. DHCP [IETF-RFC1531 1993] or) looking for MC Services tagged in their descriptions with a specific group label. If the discovered MC Service is not tagged with a group label, the Controller Component can use some mechanism provided to allow subscriptions to a generic 'welcome' group (e.g. in JXTA implementations [SUN-JXTA 2001]).

In this case, the MC Service should send a request via the MMI Status Event to the Advertise Manager subscribing to the register of the 'welcome' group. [**Figure 2.2.7**] After the bootstrapping mechanism, the Controller Component can ask for a description address and the *media* list address.

In other implementations the MC Service can send a description and the *media* list directly to the registry of the 'welcome' group during bootstrapping.

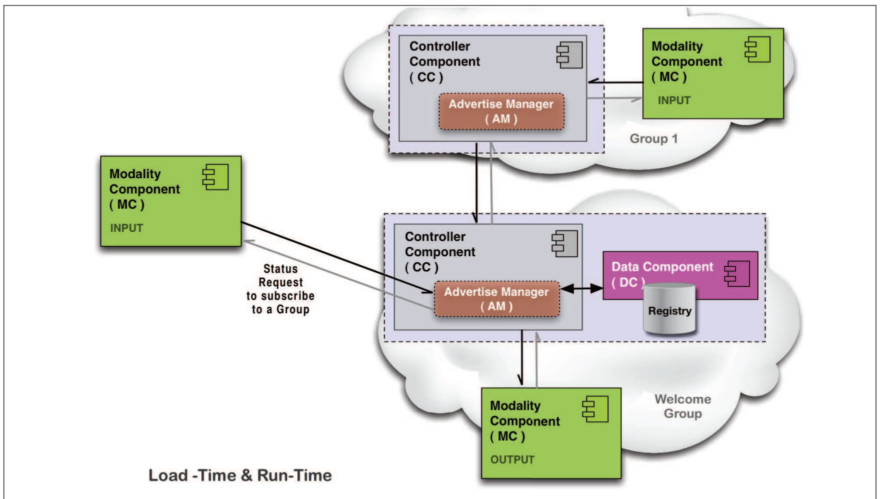

Controller Component (CC)
Advertise Manager (AM)
Modality Component (MC)
INPUT
Group 1
Modality Component (MC)
INPUT
Status Request to subscribe to a Group
Controller Component (CC)
Advertise Manager (AM)
Data Component (DC)
Registry
Modality Component (MC)
OUTPUT
Welcome Group
Load -Time & Run-Time

Figure 2.2.7: Mediated Discovery of MC Services: Federated groups

- In **active discovery**, the Controller Component can initiate a multicast, anycast or broadcast request depending on the underlying networking mechanism. This should be done via the Status Event or the Extension Notification defined by the MMI Architecture.

- The active discovery is performed by initiating a multicast request and waits for unicast responses from MC Services within its scope.[120] Thus, active discovery finds MC Services by sending requests to candidate addresses and ports and monitoring its response. Active discovery requires participation of the target MC Service and the Advertise Manager.

- In **passive discovery**, the Controller Component can listen to advertisement messages coming from MC Service over the network in a known port. [**Figure 2.2.8**]

In this case, the Advertise Manager provides a directory service (e.g. acting as a Jini Lookup Service [SUN-JINI 1998] or the UPnP [UPnP-Arch 2008] control points), looking for the MC Service's announcements that should be published with the MMI Status Event or with theExtension Notification.

A Complex Modality Component Service is used as a mediated registry providing the addressing and group information about other distributed MC Services (A,B and C). [**Figure 2.2.8**]

120 A scope is an arbitrarily assigned string (semantically annotated) that groups a number of MC services into a collection to aid scalability. MC services may be assigned one or more scope identifiers that act as filters, limiting the Controller Component to detect only s MC services within those scopes.

These services announce their network location and scope in a multicast address, which is scanned by a «nested» Controller Component searching for other MC Services of interest in the network group to index.

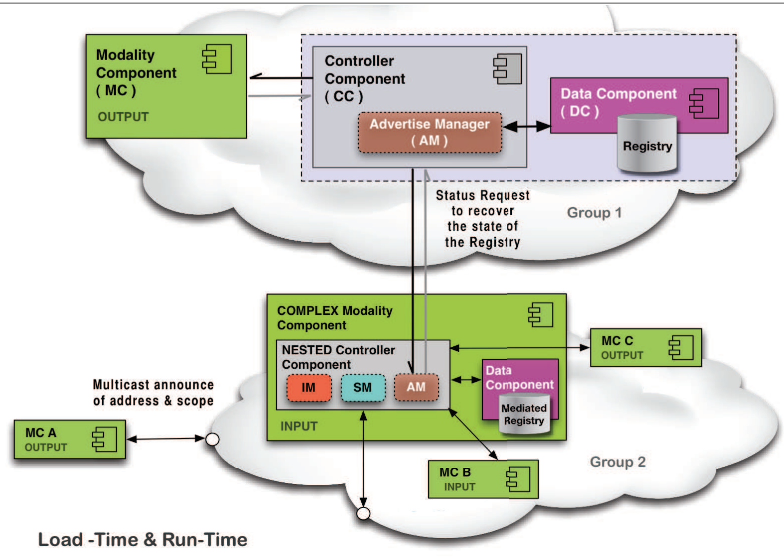

Modality Component (MC)
OUTPUT
Controller Component (CC)
Advertise Manager (AM)
Data Component (DC)
Registry
Status Request to recover the state of the Registry
Group 1
COMPLEX Modality Component
NESTED Controller Component
IM
SM
AM
Data Component
Mediated Registry
INPUT
MC C OUTPUT
Multicast announce of address & scope
MC A OUTPUT
MC B INPUT
Group 2
Load -Time & Run-Time

Figure 2.2.8: Passive Discovery of MC Services: Federated groups

Conclusion

To sum up, four types of discovery are proposed in Soa2m: MC Service intention discovery, the MC Service semantics discovery, the MC Service behavior discovery, and the MC Service capacities discovery.

And these types of discovery can be implemented with four mechanisms depending on the multimodal system needs: a fixed discovery, a mediated discovery, an active discovery or a passive discovery.

2.2.1.3 Registration

As we already presented, to improve the service discovery, semantic annotations and descriptions of non-functional characteristics (for example with a semantic markup language) can be added to service definitions in WSDL.

In this way, the use of ontologies that support shared vocabularies and domain models facilitates the service discovery by making explicit the semantics implied by the structures in the description of services.

Then, these descriptions can be stored in a registry or in multiple registries depending on the implementation.

The Soa2m registry is a service

In Soa2m a registry is a service provided by the Advertise Manager. This registry stores and disseminates the information about the service.

After the discovery phase, or during the discovery phase a Modality Component registers its description (e.g. a Capabilities Manifest or a WSDL description) in a Register Service provided by the Advertise Manager.

Then, the Controller Component or the Client Application can request service information stored in the registry in the Data Component. [**Figure 2.2.9**]

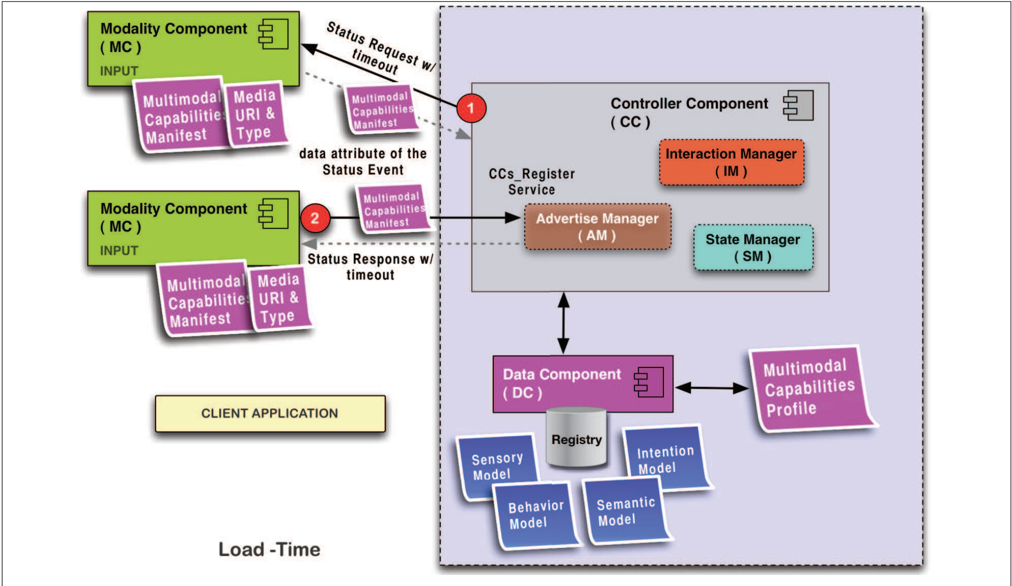

Modality Component (MC)
INPUT
Multimodal Capabilities Manifest
Media URI & Type
Status Request w/ timeout
Multimodal Capabilities Manifest
data attribute of the Status Event
1
Controller Component (CC)
Interaction Manager (IM)
CCs_Register Service
Advertise Manager (AM)
State Manager (SM)
Modality Component (MC)
INPUT
2
Multimodal Capabilities Manifest
Status Response w/ timeout
Multimodal Capabilities Manifest
Media URI & Type
Data Component (DC)
Registry
Multimodal Capabilities Profile
CLIENT APPLICATION
Sensory Model
Intention Model
Behavior Model
Semantic Model
Load -Time

Figure 2.2.9: Registering of MC Services: Push and Pull Request

The registering can be executed at design-time (hard-coded), or at run-time with a push mechanism (1) or with an adaptive pull mechanism (2). [**Figure 2.2.9**] The description is exchanged using the MMI Status Event.

Types of registration of services

The Modality Component registers its services for a specific period of time. This is the basis for the handling of the MC services state. In Soa2m we propose the life attribute to respond to this basic requirement: this is the Life attribute of the timeout triple[121].

Every MC Service has a life-time, that initiates at discovery and ends in a date provided at registering. If the Modality Component does not re-register the service before this lifetime expires, the service it is purged out. This depends on the parameters given by the Application logic, the distribution of the Modality Components or the context of interaction.

When the lifetime has no end, it is called «hard-state registering», and the MC Service is part of the multimodal system indefinitely. In «soft-state registering», in contrast, a limited life-time is associated to the MC Service, and if it is not renewed before expiration, the MC Service will be assumed no longer been part of the multimodal system. Then, by the use of a soft-state registering, the multimodal system can implement a procedure to confirm its global state and update the «inventory» of the components that could eventually participate into the interaction cycle. Therefore, soft-state and hard-state registering involves the MC Service timeout information, which is always exchanged between components and, in the case of soft-state registering, is updated from time to time.

For this reason, a registration renewal mechanism is needed. In Soa2m the mechanism of renewal is proposed with two new attributes extending the MMI Framework &Architecture: the checkUpdate Event and the UIUpdate Notification[15], used in conjunction with an automatic update process that ensures periodical requests. Explicit de-registration is also proposed with the use of the Status Event.

Proposal: Two events to support the update of registration data

As described above, the Register service provided by the Advertise Manager stores the information about the service in cache or in a registry provided by the Data Component. Each MC Service register item has its own data structure in the registry, filled with some very basic information (an unique identifier and a «intentional scope» name) or a more complete data, enhanced this basic information with four facets of metadata: **sensorial**, **behavioral**, **semantic** and **intentional**.[122]

121 See supra section 2.1.3.4 Messaging Protocol in Soa2m.
122 For a detailed description of the proposed data structures see infra section 2.2.3. Semantic Description of Multimodal Data.

Proposal: Naming convention for discovery

The basic «intentional scope» name is a text chain concatenating three tags each one with the first letter in uppercase. This name is based on the global intention of interaction to which the MC Service declares to adhere at the moment of registering, a specification of its behavior and a generic information about its media type. [**Definition 2.2.1**]

HtmlStatelessCommander

--- M -----
--------- S ----------
------------- R --------------

Where

M is the implementation media type

S is the specification of the more important attribute that differentiates the MC

R is the most important intentional role of the MC

Definition 2.2.1: The intentional scope naming convention.

The Modality Components and the Advertise Manager communicate as follows:

- Each Modality Component sends out or provides an bootstrapping message to advertise its presence on the network and its services. The fundamental exchange in both cases is a message in the form of an MMI Status Event containing a few, essential specifics about the Modality Component, its «intentional scope» name, its location, some metadata and a pointer to more detailed information.
- After the discovery on the network, the basic information about the Modality Component is registered through the Register service provided by the Advertise Manager. If more detailed information is available (e.g. a manifest or a WSDL file) it must be found, parsed and added to the registry.
- The Register service store this information in a registry in the Data Component and eventually, in its local cache. If the Data Component's implementation follows the Smart DC pattern, the Modality Component's information is processed according to the knowledge models in order to semantically annotate the declared attributes. For example, the «intentional scope» name tags are parsed and the service is annotated with the corresponding semantics. If more detailed information is provided, this metadata is matched with the models used by the multimodal system or annotated with information extracted from other semantic sources like on-line Knowledge Bases.

This communication model is based on the interaction life-cycle events of the MMI Architecture and the two extensions proposed in Soa2m.

To store the registry, the Data Component can be used as a centralized or as a distributed support.

A centralized registry hosted by a Data Component provides information about the state of MC Services and their descriptions. This has the potential risk of generating problems associated with having a centralized registry such as a single point of failure, or bottlenecks. To reduce this risk, the Soa2m architecture being implemented can use redundancy at the internal level (e.g. sharing data and responsibilities between nested components) or at the system level (e.g. backup servers).

On the other hand, a distributed registry can be hosted by multiple nested Data Components. In this case, the Data Components can be federated into groups.[123]

123 The detailed description of the distributed implementation in federated groups is out of the scope of this document. Nevertheless we introduce the problem in an spirit of completeness for the proposal, and because this is a subject to take into account for the specification of the requirements and interfaces of Data Components in the MMI Framework & Architecture recommendation which is a task to be done in the charters to come.

This is a consequence of the structure of the Soa2m Architecture: the distribution of Data Components in nested Modality Components facilitates the implementation of a multi-node registry. However, conducting inquiries across the federated environment of Data Components can be time consuming and there is the risk of some inconsistencies.

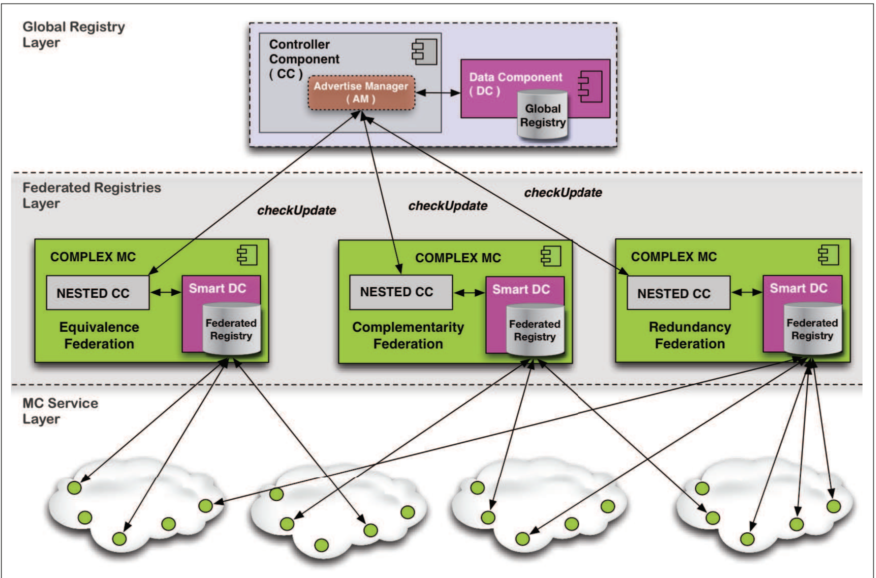


Figure 2.2.10: Federated Registries and the CARE properties

In Soa2m the inconsistency and the balance of the registry load can be resolved by organizing the data in semantically adapted federations and in consequence, reducing the requesting time. [**Figure 2.2.10**] Depending on the implementation needs, the data can be distributed according the according the CARE properties or the supported *mode*, *modality* or *media,* the usage situation, the user model or the service skills or intention.

Proposal: Distribution of registries based on Multimodal Information

For example, in [**Figure 2.2.10**] equivalent MC Services are federated in the same registry organized around the CARE properties. This facilitates the selection of the most adapted equivalent modalities in a reduced time, while the redundant ones can be selected with some latency given the physical separation of the registry.

In other cases, the federation strategy can be defined based on *mode* or *modality* or situation similarities in its description metadata, depending on the turn-taking mechanism and rules implemented. In this way, acoustic modality components stored as static data, can be found more efficiently near to a specific kind of users or application services[124].

Finally, the Soa2m architecture also allows an active collection and updating of registry data with its adaptive and mixed discovery mechanisms which enhance the real-time validity of the information of services.

In this section we presented the requirements the Soa2m architecture proposal regarding the advertisement of MC services, their discovery and their registration.

Advertisement Manager Requirements

Through this enumeration we reveal some functional responsibilities of the Advertise Manager in the Soa2m architecture:

- The Register services provided by the Advertise Manager can handle the request of services, carrying data annotated using semantic web technologies or inline data in a simpler format ready to be parsed.

124 This is currently one of the services proposed by Akamai, caching static data (and registries) in servers close to the users or the transformation MC services related to some mediaTypes proposed by CloudFlare, like the Polish and Mirage treatments. See: http://blog.cloudflare.com/introducing-mirage-intelligent-image-loading

- In the most advanced cases, the Register services in the Advertise Manager is the first entry for the process of ontology matching, which is the process of determining correspondences (alignments) between advertised concepts with the knowledge models in the *multimodal system*. If detailed information is available (e.g. a manifest or an extended WSDL file) the Register service must trigger a search, the analysis of the files by a Smart Data Component and finally, the registration of the information collected.
- The Register services provided by the Advertise Manager must support the MMI Status event and the MMI Extension Notification and also the Soa2m CheckUpdate Event and the Soa2m UIUpdate Notification.
- The Register services provided by the Advertise Manager must respond providing the default timeout structure in the data attribute of its responses to allow the adaptive pull mechanism.
- The Register services provided by the Advertise Manager must verify the scope of the request and responses and by default, register the MC Service in a welcome group if no scope is predefined.
- The Advertise Manager must provide a directory service that disseminates the information about MC Services to Client Applications, Controller Components and the State Manager.
- The Register services provided by the Advertise Manager must support automatic updates with periodical requests defined in the timeout structure . It must trigger automatic de-registration when life-time ends and explicit de-registration with the use of the Status Event.
- The Register services provided by the Advertise Manager must communicate with the Data Component to store the information about the service in a registry and eventually, in cache.
- The Register services provided by the Advertise Manager may support semantically adapted federations, providing the interfaces for the indexing and querying mechanisms.

These shared responsibilities between the Advertise Manager and the Data component give us a functional context for the semantic enrichment of the service-oriented architecture and for the description of multimodal processes and data, that will be presented in the next sections.

2.2.2 The Semantic Annotation of Services

The Soa2m project covers two approaches currently used for service discovery: the industrial approach that uses hardware devices as networked services [UPnP-Arch 2008] [ECHONET-DeviceObjects 2002], and the approach on web services discovery that considers a service as a software component performing a specific functionality [OASIS-REFERENCEMODEL 2006].

Types of services addressed in Soa2m

This means that from the Soa2m perspective, a touch display can be wrapped and interpreted as a MC Service, but also a voice recognition service or a location API can be contained in a MC Service abstraction. To describe these MC Services we use three formats: a YAML description, a RFD manifest or a description with the Web Services Description Language completed with semantic annotations[125]. This section will present the annotation mechanism proposed in the Soa2m project, which is a method resulting of the use of three different technologies.

This method evolves on difficulty and expressiveness, to allow to a diverse horizon of applications to select which mechanism is the more adapted to its needs. In result we will present in the following section the three formats and illustrate them with some use cases oriented to the discovery and registering of MC services.

Outline

The first two annotation techniques will focus on functional description, while the third will be exclusively oriented to the annotation of services by the use of ontologies. As we explained in section 2.2.1.2 Discovery, in the Soa2m architecture the Modality Component Services these ontologies reflect two possible uses of services annotation: to express the functional behavior of the service with **service** ontologies; or to express the context of use of the service with **usage** ontologies.

To sum up, first, we will present a tag-based annotation technique, second, we will present a xml-based annotation technique and finally, we will present an annotation technique enhancing XML descriptions with the two kinds of ontologies presented above.

2.2.2.1 Description of MC Services with Data Structures

In Soa2M, to facilitate modality discovery, the description with YAML data structures is proposed to support the use of a controlled vocabulary. "YAML Ain't Markup Language" (YAML)[Ben-Kiki et al. 2009] is a data serialization language designed to be human-friendly, easily readable by humans and portable between programming languages. YAML matches the native data structures of agile languages: it integrates and builds upon concepts described by C, Java, Perl, Python, Ruby, RFC0822 (MAIL), RFC1866 (HTML), RFC2045 (MIME), RFC2396 (URI), XML, SAX, SOAP, and JSON. It has a consistent model to support generic tools and is designed to allow one-pass processing.

The YAML data structures

YAML directly supports both collections (mappings, sequences) and scalars which enables programmers to use their language's native data structures for YAML manipulation, instead of requiring a special document object model (DOM). Thus, YAML provides a support for application-defined types and for representing rich data structures.

YAML provides globally unique type names using a namespace mechanism inspired by Java's DNS-based package naming convention and XML's URI-based namespaces. In addition, YAML allows for private types specific to a single application and support the processing of large documents (e.g. transaction logs) or continuous streams (e.g. feeds from a machine in run-time). In Soa2m YAML was chosen as a description language for simple MC Services in known environments, because YAML's foremost design goals are human readability and support for serializing arbitrary native data structures. Thus, YAML allows for extremely readable files. This extensibility and expressiveness is ensured because YAML ventures beyond the lowest common denominator data types, at the price of requiring more complex processing when crossing between different programming environments.

[125] See Appendix 7: Semantic Web in Soa2m.

YAML can be viewed as a natural superset of JSON, offering improved human readability and a more complete information model. This is also the case in practice; every JSON file is also a valid YAML file. This makes it easy to migrate from JSON to YAML if/when the additional features are required. However, JSON and YAML have different priorities.

JSON's foremost design goal is simplicity and universality. Thus, JSON is trivial to generate and parse, at the cost of reduced human readability. It also uses a lowest common denominator information model, ensuring any JSON data can be easily processed by every modern programming environment.

YAML is primarily a data serialization language and avoids the complexities coming from other description languages, like XML, that was designed to support structured documentation. XML therefore had many design constraints placed on it that YAML save in implementations with a before-hand known structure and with less needs of description and formalism, which is for example the case of well documented RESTful services.

For example, the functional behavior of a Commander Modality Component Service. This MC Service provides a visual widget assembling the controls for a media player without saving or recovering the state of the reproduction of the target sequence. Each control is a html button, skinned with a css file, that returns a discrete command event. This can be described in YAML as follows :

```
1  IntentionalName: HtmlStatelessCommander
2  Affiliation: DiscreteController
3  Operations: {
4    Play: {
5        Input : {
6          Selection : {
7            Event : DISCRETE
8            EventType : [ Activate ]
9            Metadata: { Content-Type: { Visual: [text/html, text/css] } }
10             }
11           }
12        Output :
13          Command : {
14           Event: DISCRETE
15           EventType : [ ExtensionNotification, Start, UIUpdate, CheckUpdate ]
16           Metadata: { Content-Type: { Cognitive: [ data/boolean, text/xml ] } }
17             }
18          }
19       }
20    Stop: {
21        Input : {
22          Selection : {
23          Event : DISCRETE
24          EventType : [ Activate ]
25          Metadata: { Content-Type: { Visual: [text/html, text/css] } }
26             }
27           }
28        Output :
29          Command : {
30           Event: DISCRETE
31           EventType : [ ExtensionNotification, Cancel, UIUpdate, CheckUpdate ]
32           Metadata: { Content-Type: { Cognitive: [ data/boolean, text/xml] } }
33             }
34          }
35       }
36 }
```

The document asserts the minimum information needed to describe the service, namely, the intentional name of the service (line 1), its affiliation[126] (line 2), its operations, the required inputs and outputs, the events triggered and the types of data handled. The description can also evolve according with the complexity of the data to describe and the goals of the description. It can add some optional non-functional data to describe the situation of use related to the service, like the multimodal properties (line 37 to 44) :

126 The Affiliation is the main concept associated with the resource that specifies the generic information given by the intentional name. This can be, for example, the inheritance relationship. The affiliation is declared by the service provider and is part of the advertisement policy. It puts forward an aspect of the service that could affect its selection.

```
37  Modality: {
38     Visual : [ TEXT , GRAPHICS ]
39     Haptic : [ SELECTOR ]
40    }
41  Medium: {
42     Visual :  [ BUTTON ]
43     Cognitive : [ SEQUENCE_CONTROLLER ]
44    }
```

The document can also give a high-level description of the architectural attributes of the MC Service (lines 45 to 49) , non-functional properties needed to handle the coordination of modalities(lines 50 to 59) or the collection of user data (lines 60 to 64).

```
45 Type: {
46    Relational : OUTGOING
47    Temporal : CHRONICLER
48    Decisional : DUMB
49   }
50 Skill: {
51    Talent : LOCALIZED
52    Weakness : UNTOUCHABLE
53    Dexterity : REMOTE
54   }
55  Life: {
56    GoodFrom : 2012-02-20
57    GoodUntil : 2013-04-22
58    NotGoodAfter : 2013-04-27
59   }
60 Privacy: {
61    DataPolicy : CATEGORIAL_ANONYMIZATION
62    BehaviorPolicy : GRAPH_ANONYMIZATION
63    InferencePolicy : CONTEXTUAL_ANONYMIZATION
64   }
```

Finally the document can optionally give a high-level description of the situation of the MC Service, covering the more relevant conditions of use, the state of the environment and the context needed for each supported mode.

For example, the Commander MC Service can be described as a service that uses a display (line 66) to work in the visual mode, and a mouse, pen or a keyboard (line 67) in the haptic mode.

```
65 Situation: {
66    Visual : { R-who: [ DISPLAY ] , P-how: [ uri:SKIN, uri:PERFORMANCE, uri:QUALITY ]}
67    Haptic : { R-who: [ [MOUSE, PEN, KEYBOARD] ] }
68    Cognitive : { S-who: [ WEBSERVER ], S-how: [ STATELESS ], S-which: [ INDOORS , NEAR_REAL_TIME, CONTROL ] }
69   }
```

This is related to the kind of event (line 7 in the code snippet on the previous page) and the content-type that it can handle (line 9 in the code snippet on the previous page) as described for the «Play» operation.

The YAML description is populated with Tags

Every property is a value that is expressed as a tag (in uppercase), a data type inspired on the MIME Types[19] (line 19 in the code snippet on the previous page) [IANA-MIME 2012] or an event conforming with the MMI specification or the DOM Level 3 recommendation (line 5) [W3C-DOM3 2012].

The basic principle of tagging is that end users index subjects instead of experts only. This is a mode of description of entities based on collaborative efforts and can be formalized by a generic common vocabulary. In most tagging applications, the set of tags are unrelated and completely unstructured. To introduce a hierarchy structure by the means of a vocabulary enhances expressivity of the model and also helps to keep the descriptions succinct.

Thus, the YAML tagging in Soa2m involves two steps: a data model and a denotation[127] procedure.

127 Denotation is the process of finding an appropriate set of index terms that represent a particular facet of an entity or process.

The data model analysis involves deciding on what a resource is about and what is relevant in particular. Note that the result of this conceptual analysis heavily depends on the needs and interests of the domain that a resource is tagged for[128]. Denotation is the mechanism of finding an appropriate set of index terms that represent a particular facet of an entity or process.

Two difficulties arise in the tagging process: the pertinence for the domain of this intuitive (based on common sense) data model; and the polysemic nature of language (the multiple number of meanings, interpretations or understandings of words). Synonyms and homonyms/homographs are frequent problems in the process of denotation that a Glossary[129] in Soa2m tries to eliminate by providing a list of preferred and non-preferred terms, together with definitions and a semantic structure.

In the Soa2m architecture, this semantic metadata is defined in a generic Multimodal Data Model. The data model is a dataset of tag annotation categories to be exchanged through the attributes proposed by the MMI protocol and related to the set of ontologies described in section 2.2.3 Semantic Description of Multimodal Data.

Conclusion

To sum up, in Soa2m the Modality Component Services can be described with a YAML file that evolves on complexity depending on the application needs.

Inspired on the device descriptions[22] of Card, Mackinlay et Robertson [Card et al. 1991], this description can be limited to indications about the Input and Output interfaces of the MC Service or be more detailed describing functional and non-functional properties and information about the situation of use.

The attributes in the description are defined in a multimodal data model and a multimodal vocabulary that will be presented in section 2.2.3. Semantic Description of Multimodal Data. The values of these attributes of the YAML file can be tags, events or mime types that will be parsed by the Controller Component in the phase of discovery in order to populate a registry.

2.2.2.2 Description of MC Services with RDF Manifests

The knowledge of all things is defined by Aristotle with the term «first philosophy»: the science of main causes, main principles and the purpose of all-that-is as it is. The description of this knowledge about our knowledge was called by Aristotle's editors the Meta-Physics, a discipline with three branches.

Historical origin of the concepts description, interpretation, intention

First, the study of these principles (the How) called the Aristotle's science of being; the study of the categories of things and ideas and its relations (the What) called the Aristotle's ontology; and the study of the causes that explain these properties (the Why) called the Aristotle's theology.

Thus, in the Aristotle approach, to access to the knowledge about the world, three facets must be addressed: 1) the abstract relations between things, 2) the categories of things (and concepts) and 3) the intention -causes- behind the existence of these relations and things.

In some symmetric way, in the contemporary semantic web, the knowledge about the human knowledge is generally called metadata, and **ontologies** are the effort to organize this human knowledge in a machine-understandable way:

> *«Metadata is machine understandable information about web resources or other things The phrase "machine understandable" is key. We are talking here about information which software agents can use in order to make life easier for us, ensure we obey our principles, the law, check that we can trust what we are doing, and make everything work more smoothly and rapidly. Metadata has well defined semantics and structure.*

128 For a detailed description of the proposed data model see infra section 2.2.3. Semantic Description of Multimodal Data.
129 See infra section 2.2.3.1. A Glossary for Multimodal Discovery.

Metadata was called "Metadata" because it started life, and is currently still chiefly, information about web resources, so data about data. In the future, when the metadata languages and engines are more developed, it should also form a strong basis for a web of machine understandable information about anything: about the people, things, concepts and ideas.» [Berners-Lee 1997]

Ontologies in computer science

Thus, in computer science, ontologies describe the formalization of a knowledge. They are explicit specifications of a conceptualization and the product of a common agreement of a group or community of people about concepts, categories, relation between concepts, properties that describe the concepts, axioms and constraints.

As we presented above[130], there are different approaches for modeling ontologies such as software design technologies (UML), database technologies (Entity-Relationship diagrams), and Artificial Intelligence based techniques (Description Logics, Frames and First Order Logics).

The formalism adopted to model an ontology constrains the possible software implementations and the reuse of the represented knowledge. It also restricts the way of defining concepts and the constraining relationships between concepts.

In this section we will present the second format of description of MC Services proposed in the Soa2m architecture.

This format serves primarily for formal description of concepts and relations (roles). Semantically, this description have good computational properties such as decidability in restricted conditions[131] and is found on predicate logic[132].

Formal tools used in the Soa2m ontologies

In our description of services based on ontologies, we use Artificial Intelligence (AI) techniques to model them, like the Frames and First Order Logics, and Description Logics (DL) combined with weak AI techniques, like the situational approach.

In Soa2m the frames represent concepts described through predicates (classes, properties and relations) and axioms defining rules that restricts their relationships while Description Logic functions describe binary relations between concepts.

In these kinds of representations it is possible to create axioms and make inferences with them, for example, if a Controller Component is a nested component, it is possible to infer that there exists some Complex Modality Component who contains it.

Technical tools used in the Soa2m ontologies

Soa2m proposes the use of the semantic web technologies to represent knowledge in this current second format of description for Modality Component Services.

Semantic web technologies support the creation of these representations in Soa2m using some of the languages illustrated in [**Figure 2.2.11**].

130 See supra section 1.2.3 The Interaction Context of a Multimodal System

131 A proposition is decidable if a membership in its set of formulas (or theorems) can be effectively determined. In computer science, a problem is decidable if there is a mechanical procedure that terminates in a finite number of steps, which allows to respond yes or no to a question raised by the problem.

132 Propositional logic is an axiomatization of the Boolean logic. Propositional logic is decidable and complete, for example using the method of truth tables. On the other hand, predicate logic (also called predicate calculus and first-order logic) is an extension of propositional logic to formulas involving terms and predicates. For example, the assertion «x is greater than 1», where x is a variable, is not a proposition because you can not tell whether it is true or false unless you know the value of x. Thus the propositional logic can not deal with such sentences. In result, the predicate logic is an extension of propositional logic that also has variables for individual objects, quantifiers, symbols for functions, and symbols for relations envisioned to cover such assertions that appear quite often in mathematics. With the predicate logic it is possible, in some restricted conditions, to do inferencing on those assertions even if the full predicate logic is undecidable. In other words, predicate logic allow us to generate conclusions about incomplete knowledge.

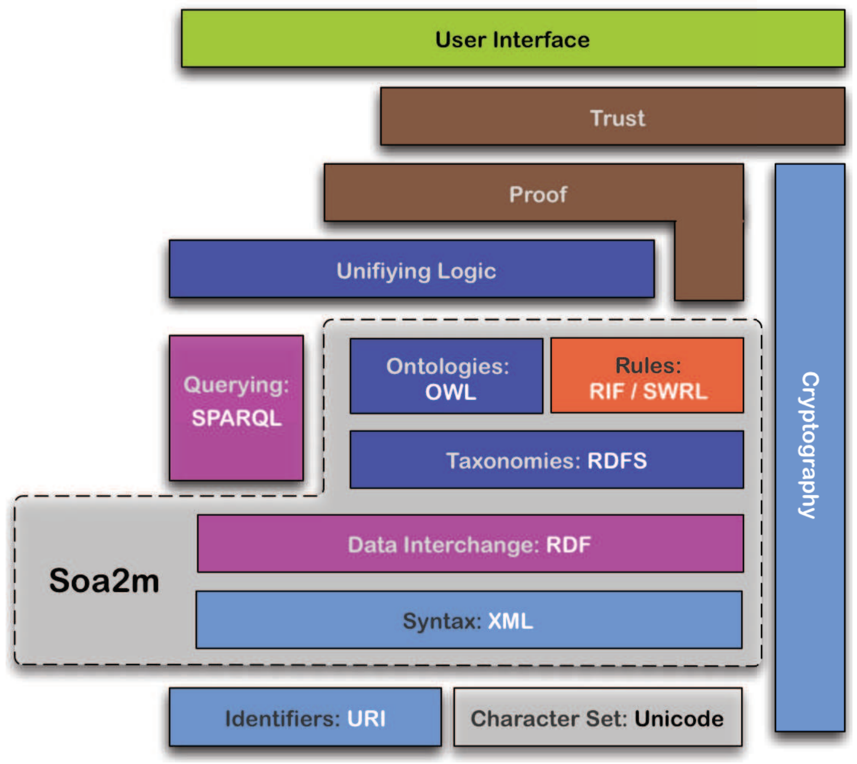


Figure 2.2.11: The Semantic Web Layers used in Soa2m

Proposal: A method of description based on a faceted thesaurus

In Soa2m, the YAML format of tagging to describe Modality Component Services, is followed by a description based on a Faceted Thesaurus[29] of semantic descriptions using these layers.

This description mode is inspired on the generic Interaction Tasks of Foley [Foley et al. 1980] in which generic classes of virtual devices, such as «Locators» and «Selectors» are used to described interaction cycles.

The goal is to permit the easy substitution of one physical device for another of a similar class (or the same class).

This facilitates modality discovery, finding the best among the alternatives but with an important limitation: the technical interchangeability of these devices does not necessarily ensure usability [Buxton 1986].

In Soa2M, to allow modality discovery, the description with RDF sentences is proposed as a method to annotate the services provided by Modality Components.[133]

We propose the description od Modality Component Services using RDF manifests. These manifests enumerate the MC Service metadata in the context of semantic web services as proposed in Soa2m.

The following code is an example of a RDF/RDF-S manifest for the FlexVoiceSynthesizer MC Service.

First, we create somme tiny URIs (line 1 to 7) for the semantic and behavioral domain descriptions, then we link the document to the semantic web technologies (line 8 to 9) and finally, we declare the document namespaces for some important classes (in RDF-S) and descriptions, to enhance readability (line 10 to 15):

133 For an Introduction of RDF and RDF-S Technologies see infra Appendix 7 Semantic Web in Soa2m

```
1  <?xml version="1.0"?>
2 <!DOCTYPE rdf:RDF [ <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">
3 <!ENTITY xm "http://localhost/soa2m/xs/Soa2MSchema#">
4 <!ENTITY soa2m "http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/architecture/">
5 <!ENTITY SemanticDescription "http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/">
6 <!ENTITY BehavioralDescriptionModality "http://localhost/soa2m/ont/participants/domain/modalities/functional/">
7 <!ENTITY BehavioralDescriptionMedia "http://localhost/soa2m/ont/participants/domain/application/functional/"> ] >
8  <rdf:RDF xmlns:rdf   = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
9     xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
10    xm:base  = "http://localhost/soa2m/ont/participants/domain/"
11    xmlns:MCs = "http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/architecture/MCs.rdfs#"
11    xmlns:Mode = "http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/Mode.rdfs#"
12    xmlns:Event = "http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/architecture/Event.rdfs#"
13    xmlns:Media = "http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/Media.rdfs#"
14    xmlns:MediaDescription = "http://localhost/soa2m/ont/participants/domain/application/functional/Media.rdf#"
15    xmlns:Situation = "http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/architecture/Situation.rdfs#" >
```

Then we announce that the subject of the statement (line 16) is identified with the code MC_1234 and the name FlexVoiceSythesizer. Then we declare that the resource being described is associated to a more generic class Synthesizer#Voice (line 17) described functionally in a Soa2m ontology.

```
16    <rdf:Description rdf:about = "http://localhost/soa2m/desc/rdf/MC_1234.rdf#FlexVoiceSynthesizer">
17    <rdf:type rdf:resource = "&BehavioralDescriptionModality;continous/Synthesizer.rdf#Voice" />
```

After the identification process, we declared that as the linked RDF schema affirms - identified by the shortcut «MCs» -, this resource has an intentional name (line 18). In other words, this MC Service inherits the property «hasIntentionalName» from a related «MCs» class.

Then we proceed to describe the instantiation of this property, in our current case, the FlexVoiceSynthesizer MC Service. The property value, **hasIntentionalName** is a composition (line 18) of three ordered informations as we described in section 2.2.1.3 Registration: Media (line 20 to 22), Specification (line 23 to 25) and Role (line 26 to 28).

The RDF description allows us to affirm that the Media structure is a functional information (line 20) while the other two are semantic and more generic informations (line 23 and line 26). It also declares that the Media value «Flex» is related to the extension of the concept Framework, which in Soa2m is considered as functioning with discrete entities[134].

For the Role value, it declares that the FlexVoiceSynthesizer MC Service is related to the concept Synthesizer, in other words, that its principal role is to act as a Synthesizer. And finally, it declares that its role as a Synthesizer is specified with the Voice concept, in other words, its activity is at least, limited to the category of Voice generation.

```
18 < MCs:hasIntentionalName >
19          < rdf:Seq >
20                <rdf:li><rdf:Description rdf:about = "&BehavioralDescriptionMedia;discrete/Framework.rdf#Flex" >
21                      <rdf:type  rdf:resource = "&soa2m;IntentionalName.rdfs#Media"  />
22                </rdf:Description></rdf:li>
23                <rdf:li><rdf:Description rdf:about ="&SemanticDescription;Voice.rdfs#Voice" >
24                      <rdf:type  rdf:resource = "&soa2m;IntentionalName.rdfs#Specification"  />
25                </rdf:Description></rdf:li>
26                <rdf:li><rdf:Description rdf:about = "&SemanticDescription;Synthesizer.rdfs#Synthesizer" >
27                      <rdf:type  rdf:resource = "&soa2m;IntentionalName.rdfs#Role"  />
28                </rdf:Description></rdf:li>
29             </ rdf:Seq >
30 </ MCs:hasIntentionalName >
```

134 In Soa2m, a discrete entity is an endurant: an entity that is wholly present at any time they are present, in contrast with continuous (perdurant) entities that extend on time accumulating its different parts. For example, a song singed by a voice is a perdurant while the person singing is an endurant. The song is wholly present only as an experience (each moment T of the song is not the whole song) while the person is completely present at any moment T while singing, even if some properties of the person can change, for example, the person can start to sing seated and then, get up during the song to continue to sing in stand up position. In conclusion, for the Soa2m purposes, the song is a continuous entity named perdurant and the singer is a discrete entity called endurant, that participates in the song. For more information about Soa2m endurants see infra section 2.2.3. Semantic Description of Multimodal Data.

Another example of the advantages of this format of description is the restriction of types, for example., for the operations provided by the Synthesizer MC Service[135]:

```
61 <rdf:li><rdf:Description rdf:about = "&BehavioralDescriptionModality;continous/Synthesizer.rdfs#Start" >
62       < MCs:hasInput >
63            < rdf:Seq >
64                 <rdf:li><rdf:Description rdf:about ="&BehavioralDescriptionMedia;continous/Synthesizer.rdfs#Stream" >

65                      < MCs:hasEvent rdf:resource = "&soa2m;Event.rdfs#Continous"  />

66                      < MCs:hasEventType >
67                             <rdf:Description>
68                                  <rdf:value rdf:datatype  ="&xm;EventList" >
69                                      [ ExtensionNotification, UIUpdate, Start, Resume, Compositionstart, Compositionupdate ]
70                                  </rdf:value>
71                             </rdf:Description>
72                      </ MCs:hasEventType >

73                      < MCs:hasMetadata >
74                             <rdf:Bag>
75                                   <rdf:li><rdf:Description rdf:about ="&soa2m;Metadata.rdfs#Metadata" >
76                                       < MCs:hasContent-Type >
77                                             <rdf:Bag>
78                                                  <rdf:li><rdf:Description rdf:about ="&soa2m;Mode.rdfs#Cognitive" >
79                                                     < Event:hasDataType >
80                                                        < rdf:Seq >
81                                                        <rdf:li>
82                                    <rdf:Description rdf:about ="&BehavioralDescriptionMedia;discrete/Framework.rdfs#Html" />
83                                                        </rdf:li>
84                                                        <rdf:li>
85                                    <rdf:Description rdf:about ="&BehavioralDescriptionMedia;discrete/Framework.rdfs#Xml" />
86                                                        </rdf:li>
87                                                        <rdf:li>
88                                    <rdf:Description rdf:about ="&BehavioralDescriptionModality;discrete/Structure.rdfs#Json" />
89                                                        </rdf:li>
90                                                        <rdf:li>
91                                    <rdf:Description rdf:about ="&xm;string" >
92                                                        </rdf:li>
93                                                        </ rdf:Seq >
94                                                     </ Event:hasDataType >
95                                   <rdf:Description> </rdf:li>
96                                             <rdf:Bag>
97                                       < MCs:hasContent-Type >
98                                   </rdf:Description> </rdf:li>
99                             </rdf:Bag>
100                     </ MCs:hasMetadata >
101                </rdf:Description> </rdf:li>

102                <rdf:li><rdf:Description rdf:about ="&BehavioralDescriptionMedia;continous/Synthesizer.rdfs#Language" >
103                     <rdf:type ="& soa2m;Language.rdfs#Prefix" >
104                     <rdf:value  rdf:datatype  ="&xsd;string" > en </rdf:value>
105                </rdf:Description> </rdf:li>
106           </ rdf:Seq >
107       </ MCs:hasInput >
108 </rdf:Description  ></ rdf:li>
```

This description affirms that the operation receives continuous data as input (line 65), that its input arguments are an ordered set (with the <rdf:Seq> tag in lines 62 to 63) or that the type of events must be interpreted according with an external data type schema (line 68 to 70). It also affirms that the metadata and the content-type are a non-ordered sets of information (with the <rdf:Bag> tag in lines 73/74 and 76/77) while the supported dataTypes for events are ordered according with some semantics (with the <rdf:Seq> tag in lines 79/80).

135 As an example we will only present the description of the Start operation.

As we can see the level of abstraction covered by the description will depend on the application needs. The description can be limited to declare the dataTypes allowed for the properties in order to control the vocabulary values, or to be extended to semantic relations like the equivalence and the hierarchy. This will depend on the demands of expressivity on the description required by the application.

Descriptions with different levels of complexity depending on the final needs

In addition, in the Soa2m architecture we propose to extend the manifest to non-functional data for some context-awareness. Depending on the application needs the manifest can include more detailed information about the situation of use[136] :

```
284 < MCs:hasSituation >
285       < rdf:Bag >
286           <rdf:li><rdf:Description rdf:about = "&soa2m;Mode.rdfs#Acoustic" >
287               < Situation:hasWhoActor >
288                   < rdf:Seq >
289                       <rdf:li>
290                           <rdf:Description rdf:about = "&BehavioralDescriptionMedia;continous/Transducer.rdfs#Speaker" />
291                       </rdf:li>
292                   </ rdf:Seq >
293               </ Situation: hasWhoActor>
294           </rdf:Description></rdf:li>
295       </ rdf:Bag >
296 </ MCs:hasSituation >
```

In this example, we inform that in order to use the Synthesizer MC service, the situation of usage from an acoustic mode perspective (line 286), must include a participant described with the hasWhoActor property (lines 287 to 293). This situation is described taking into account a behavioral facet (line 290) and affirms that this participant is a Speaker, which is an individual of the class Transducer (Transducer.rdfs#Speaker) that acts in a continuous way.

In this section we presented on detail the second format of description for the Modality Component Services which is intended to express a Faceted Thesauri of Multimodal terms.

Conclusion

This terms describe at least the IOPE informations of the service in a range going from the datatypes needed on inputs and outputs, to the relations like MC Services Equivalency or Hierarchy.

The description format is selected to integrate descriptions with the knowledge provided by the semantic web, using RDF statements and RDF-S schemas in a semantic Manifest document associated to a Modality Component Service.

2.2.2.3 Services Description with WSDL Documents

Web services technologies provide reliable software interoperability across devices, platforms and networks; and they promote the use of semantically well-founded reasoning about processes. The building block of SOA-based solutions is the self-describing Web service that can be reused across various applications.

WSDL was created specifically for this purpose and describes the data elements, interfaces, operations and bindings. [**Figure 2.2.14**] WSDL is an XML-based language that is used to describe the functionality offered by a Web service. It specifies a mechanism of encoding and a protocol for Web Service providers to describe the means of interacting with the provided services. WSDL describes network-reachable services and maps these to communication endpoints. WSDL separates the abstract definition of handled types and interfaces from their concrete binding to a network endpoint address.

Quick introduction to WSDL in Soa2m

However, WSDL descriptions are not sufficient to unambiguously decipher each operation process and its requirements.

136 As an example we will only present the description of one aspect of the situation.

Proposal: To use SAWSDL to extend with ontologies a WSDL file

To overcome the limitations in the requirements description, the OWL for Services (OWL-S) recommendation can be used[33], because it provides a model for defining processes, entities and its relationships; and is designed to enable users and software agents to automatically discover, invoke, compose, and monitor services resources.

This is why, in Soa2m we use the Semantic Annotations for WSDL and XML Schema (SAWSDL) recommendation of the W3C to overcome the ambiguity and enhance expressivity as we already explained in section 2.2.1.2 Discovery.

By adding meta-data to the WSDL elements using the *modelReference* attribute a link to a concept in an ontology is added. And this ontology will then define the semantics of the annotated element.

In Soa2m we propose to add OWL-S metadata[137] to express the higher level knowledge about the functional information «binded» by the WSDL[138] and DOLCE-Lite [Gangemi et al. 2002] metadata to express the knowledge about the context of usage of the Modality Component Service.

It is also used to model, annotate, describe, register and handle the contextual information stored in the MMI Data Components.

In the next sections we will describe the OWL-S annotation method that we propose followed by the annotation method based on DOLCE-Lite ontologies.

The intention been to illustrate an annotation mechanism that will allow to express the multimodal semantic data. This multimodal semantic data will be described later in a further section.

In Soa2m we propose the use of OWL and OWL-S in this third format of description, for applications with the need of more expressivity to represent the data model.

This format is used in Soa2m to express knowledge about data structures, for example, in the description of the requirements related to the registering of Modality Component Services.

With this format we can affirm information like:

- the changes of state implied by the successful registering,

- the mandatory importance of the **MC_code** argument as part of the **MC_registerData** (lines 55 to 59)

- or the fact that the **MC_registerData** must have at least one **MC_code** (lines 60 to 68) to consider a Modality Component service as part of the *multimodal system*.

137 For an Introduction of OWL-S Technology see infra Appendix 7 Semantic Web in Soa2m

138 Another well known effort to provide high level information about services is the Web Services Modeling Ontology (WSMO) [W3C-WSMO 2005]. Nevertheless, OWL-S is more mature in certain aspects, including the choreography and grounding specifications while WSMO provides a more complete conceptual model as it addresses non-functional aspects such as goals and mediators. OWL-S does not separate what the user wants from what the service provides. In WSMO, a Goal specifies what the user wants and the Web service description defines what the service provides through its capability. OWL-S defines only one Service Model per service, hence there is only one way to interact with the service. WSMO allows the definition of multiple interfaces for a single service.

Even if WSMO is more complete it is also more complex to implement in simple use cases, which increases the learning difficulty on a subject that is already difficult to understand. And as its authors affirms, it is also less mature. For these reasons we decide to use OWL-S instead of WSMO, and to propose the use of the modelReference attribute and the WSDL 2.0 standard to resolve the limitation of multiple interfaces and goal-oriented description in web services. In addition, using the modelReference, we can link the WSDL file to contextual ontologies defined and modeled in an independent way.

In Soa2m we decide to clearly separate the context aspects to use a more adapted and expressive set of knowledge that the ontology languages coming from the web services community. In a certain way, it corresponds to real-life multi domain collaborations, in which designers and social sciences experts complete the technical proposals in a collaborative way. For us, the domain knowledge is conceptually different and structured in a different way than the functional knowledge, and this justifies to separate both concerns on the description of web services, leaving to the final user the freedom to choose how to describe the context of usage that its service targets.

```
<owl:ObjectProperty rdf:ID="hasCode">
      <rdf:type rdf:resource="&owl;#FunctionalProperty"/>
      <rdfs:domain rdf:resource="#MC_registerData"/>
      <rdfs:range rdf:resource="#MC_code"/>
</owl:ObjectProperty>

<owl:Class rdf:ID="withoutCode">
      <owl:intersectionOf rdf:parseType="Collection">
          <owl:Class rdf:about="#MC_registerData" />
             <owl:Restriction>
                 <owl:onProperty rdf:resource="#hasCode"/>
                 <owl:maxCardinality rdf:datatype="&xsd;#integer"> 1 </owl:maxCardinality>
             </owl:Restriction>
      </owl:intersectionOf>
</owl:Class>
```

In addition, in Soa2m the third format of description uses OWL-S in conjunction with SWRL (Semantic Web Rule Language) [W3C-SWRL 2004] to describe conditional processes. The SWRL (Semantic Web Rule Language) is a rule language that combines OWL with the Datalog rule markup language.[139]

With the use of this rule language changes in the «world» state can be used in the MC services composition (or in Modality *fusion/fission*) to determine the satisfiability of the preconditions in the other MC services that participate to the multimodal interaction cycle.

Justification of the use of DOLCE

However, the context of usage of the service can also be affected by the service result, for example, the acoustic conditions in the room can change when the Start operation is executed in a SoundPlayer MC using Speakers. The multimodal application can need this information if the goal is to reproduce sound in a private manner: in this case the functional information is not enough. To overcome this limitations in Soa2m we propose the use of DOLCE-Lite ontologies.

- **DOLCE** is a library of upper ontologies[140] [Gangemi et al. 2002] written on OWL.[141] DOLCE has a cognitive orientation: it aims at capturing the ontological categories underlying natural language and human common sense. This characteristic is crucial to our goal, which is the description of the interaction phenomenon (a continuous perceptive experience) from a multimodal perspective (a situation of colocation of different entities). By using the DOLCE approach we will be capable to describe for example, the *mode* perception, the user activity, the interaction exchanges, the colocation of modalities or the situation of usage from a common sense point of view.

More precisely, in Soa2m the use of this foundational ontology is justified by its utility in numerous aspects regarding multimodality. For example:

- In Soa2m the user-interaction is viewed as a *cycle,* a continuous and symmetric phenomenon linking multiple inputs and outputs.
- In Soa2m the multimodal interaction is a multidimensional phenomenon in which co-location and simultaneity are the most important attributes.
- In Soa2m the multimodal interaction must be described as a set of events of diverse granularities.
- In Soa2m present, past and future information is needed to decide the best multimodal situation to compose with the best resources available.
- Interaction data is usually based on continuous streams of events. In Soa2m we need knowledge structures capable to represent this phenomena with well founded categories.

139 For an Introduction of SWRL Technologies see infra Appendix 7 Semantic Web in Soa2m

140 An upper ontology is an ontology that describes very general concepts that are the same across all knowledge domains. Is a conceptual reference frame to which a specific (domain) ontology can adhere.

141 For an Introduction of DOLCE ontologies see infra Appendix 7 Semantic Web in Soa2m

As discussed in section 2.2.1.2 Discovery, in Soa2m, the third format of metadata annotations is made using semantic web technologies. In this format, ontologies are linked to the service description written in WSDL.

The *modelReference* attribute is used to add situation annotations to WSDL elements.

It relies the description with **usage** ontologies based on the DOLCE-Lite and D&S plugin ontologies. These ontologies are represented in Soa2m by the prefix « **use** ».

For example, in the following code snippet line 91, the service element is linked to an ontology:

sawsdl:modelReference="&use;continuous/**Listener.owl#myServicesListener**"

This «Listener» ontology describe with DOLCE-Lite what a «Listener» means in common sense ensuring the **intention discovery** of the MC service. It also expresses that for this class of Listeners, there exists one instance, called myServicesListener, that represents the current Modality Component Service.

The Listener class is in Soa2m part of a taxonomy of modalities, structures from a high level perspective and following the paradigms of classification proposed by the taxonomy of artifact functions of DOLCE [Guarino et al. 2006], the interaction tasks of Foley [Foley et al. 1980] and the taxonomy of representational modalities of Bernsen [Bernsen 1994].[142]

In line 86 the operation UPnPAvailabilityList element is linked to an ontology describing the notion of «Availability» in the domain of communication enhancing the **semantic discovery** of the MC service.

While the first ontology provides information about the advertised intention behind the service (to provide a service to monitor services) the second ontology provides high levels details about the semantics of a precise operation in a precise domain (to monitor communication availability). In this way, both address not how the service works, but what the service does in a defined state of affairs (a context of usage).

```
75  <wsdl:interface name="ServicesListenerInterface" >
76     <wsdl:operation name="UPnPAvailabilityList" pattern="http://www.w3.org/ns/wsdl/in-out"
         sawsdl:modelReference="&ser;discrete/Listener#UPnPAvailabilityList" >

77          <wsdl:input messageLabel="UPnPAvailabilityListInput"
              element="tns:UPnPAvailabilityListRequest" />
78          <wsdl:output messageLabel="UPnPAvailabilityListOutput"
              element="tns:UPnPAvailabilityListResponse" />
79          <wsdl:documentation>
80              This is the Availability Listener Service oriented to UPnP networks.
81              It provides a List of UPnP services available at the timestamp of the request.
82          </wsdl:documentation>
83     </wsdl:operation>
84  </wsdl:interface>

85  <wsdl:binding name="ListenerSOAPBinding" interface="tns:ServicesListenerInterface"
       type="http://www.w3.org/ns/wsdl/soap"
       wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/" >

86     <wsdl:operation ref="tns:UPnPAvailabilityList"
         wsoap:action="tns:MCs_GpacServicesListener#UPnPAvailabilityList"
         wsoap:mep="http://www.w3.org/2003/05/soap/mep/request-response"
         sawsdl:modelReference="&use;discrete/Availability.owl#myCommunicationAvailability />

87     <wsdl:documentation>
88     This is the SOAP Binding for the Availability Listener Service.
89     </wsdl:documentation>
90  </wsdl:binding>

91  <wsdl:service name="MCs_GpacServicesListener"
      interface="tns:ServicesListenerInterface" sawsdl:modelReference="&use;continuous/Listener.owl#myServicesListener" >
92    <wsdl:endpoint name="ServicesListenerEndpoint" binding="tns: ServicesListenerSOAPBinding"
        address="tns:MC_5678/MCs_GpacServicesListener.php" />
93  </wsdl:service>
```

142 See infra section 2.2.3. Semantic Description of Multimodal Data

On the other hand, the *modelReference* attribute also relies the description with **service** ontologies based on OWL-S and represented in Soa2m by the prefix « **ser** » (line 76). These ontologies are a support for the **behavior discovery** with formalized and well described processes and outcomes.

And finally, the WSDL file allows the **capacities discovery**, to express the formalized dataTypes annotated with semantic non-functional annotations (lines 25 of the code snippet in the next page) from with a high level perspective, and completed (if needed) by contextual knowledge.

For example, line 25 references a service ontology that can describe what kind of parameters are needed in the request and what are the relationships and restrictions for these parameters from a contextual point of view but also what means each parameter in the current context and in the current domain (lines 39 to 41).

In this way, a parser can infer that if the parameter «serviceHostName» is filled, the result will reflect some kind of location of the services according to a specific *state of affairs*.

In this case, the location is related to the UPnP protocol and correspond to a network address; in other cases it can represent the physical space in which the service can be provided, for example, the coffee room.

This will depend on the situation description for this service.

As an illustration of these high level annotations, we can present the « use » ontology linked on line 91 of the code snippet in this page:

« &use;continuous/Listener.owl#myServicesListener ».

```
21 <wsdl:types>
22   <xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
23    targetNamespace="http://localhost/soa2m/desc/wsdl/" >
24      <xs:element name="UPnPAvailabilityListRequest"
25       sawsdl:modelReference="&ser;discrete/Requester#myParametrizedRequest"
26       type="tns:checkUpdateRequestCT">
27         <xs:complexType name="checkUpdateRequestCT">
28            <xs:sequence>
29               <xs:element name="context" type="mmi:context.optional.attrib"/>
30               <xs:element name="source" type="mmi:source.attrib"/>
31               <xs:element name="target" type="mmi:target.attrib"/>
32               <xs:element name="type" type="mmi:type.attrib"/>
33               <xs:element name="requestID" type="mmi:requestID.attrib"/>
34               <xs:element name="updateType" type="soa2m:updateType.attrib" />
35               <xs:element name="state" type="soa2m:state.attrib"/>
36               <xs:element name="data" minOccurs="0" type="mmi:anyComplexType">
37                  <xs:complexType name="UPnPAvailabilityListParameters"
                     sawsdl:modelReference="&use;discrete/Listener#myAttentiveListener">
38                    <xs:sequence>
39                       <xs:element name="onlyTotal" sawsdl:modelReference="&use;discrete/Focus#myQuantity" />
40                       <xs:element name="serviceType" sawsdl:modelReference="&use;discrete/Focus#myType"/>
41                       <xs:element name="serviceHostName"sawsdl:modelReference="&use;discrete/Focus#myLocation" />
42                    </xs:sequence>
43                  <xs:complexType
44               </xs:element>
45            </xs:sequence>
46         </xs:complexType>
47      </xs:element>
```

This is an OWL document describing the non-functional properties of the MCs_GpacServicesListener service, which is a service providing information about the state of other Modality Components taking into account some multimodal data described as a situation.

In this way, this service participates in the tasks concerning context-awareness and state management in a multimodal system constructed according with the Soa2m architecture.

```
1   <rdf:RDF xmlns="&listener;"
2     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
3     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5     xmlns:owl="http://www.w3.org/2002/07/owl#"
6     xmlns:xm="http://localhost/soa2m/xs/Soa2MSchema#"
7     xmlns:DOLCE-Lite="http://localhost:8888/soa2m/ont/srcs/DOLCE/DLP/DOLCE-Lite.owl#"
8     xmlns:Soa2mSituation="http://localhost:8888/soa2m/ont/srcs/SOA2M/situation/Soa2mSituation.owl#" >

9   <owl:Ontology rdf:about="&listener;">
10  <rdfs:label rdf:datatype="&xsd;string"> Listener </rdfs:label>
11  <rdfs:comment rdf:datatype="&xsd;string">
12    [ INTENTION: To Listen.l DEFINITION: An agentive person or object  play the role of Listener
      because it listens Somebody or Something (e.g. a person in a conversation, an artifact
      monitoring states, a robot receiving orders ). ]
13  </rdfs:comment>

14  <owl:imports rdf:resource="http://localhost:8888/soa2m/ont/srcs/SOA2M/situation/Soa2mSituation.owl"/>
15  <owl:imports rdf:resource="http://localhost:8888/soa2m/ont/srcs/SOA2M/mode.owl"/>
16  </owl:Ontology>

17  <owl:AnnotationProperty rdf:about="&rdfs;label"/>
18  <owl:AnnotationProperty rdf:about="&rdfs;comment"/>

19  <owl:NamedIndividual rdf:about="&listener;myServicesListener">
20    <rdf:type rdf:resource="&Soa2mSituation;Soa2mSituation"/>

21    <rdf:type> <owl:Restriction>
22        <owl:onProperty  rdf:resource="&Soa2mSituation;hasFacet"/>
23        <owl:allValuesFrom  rdf:resource="&Soa2mSituation;sensorialFacet"/>
24    </owl:Restriction> </rdf:type
25    <rdf:type> <owl:Restriction>
26        <owl:onProperty  rdf:resource="&Soa2mSituation;hasGranularity"/>
27        <owl:allValuesFrom  rdf:resource="&Soa2mSituation;relationalGranularity"/>
28    </owl:Restriction> </rdf:type>

29    <mode:analogically-references  rdf:resource="&mode;acoustic"/>
```

The ontology affirms that the Modality Component Service linked to the file exists as a continuous entity (a perdurant) described by a situation structure. The situation used to describe the service is called «myServicesListener» and it is an individual of the type Soa2m Situation (line 19 to 20).

This situation describes the perdurant from the sensorial point of view of the event (line 22). This means that the information given will focus on the sensorial facet of the phenomenon «service listening» (line 23). And the description also, will give the data in the relational granularity, which means that it will provide the information that describes the relationships appearing in the sensorial facet of the phenomenon (line 26 to 27).

Finally, the information provided by the description can be enriched by other descriptions by analogy. Analogy (in Greek "shared knowledge") is a cognitive process of transferring information or meaning from a particular subject to another particular subject. To maximize complex learning, the use of «succedaneum» (something that can be used in-place of another) is a current practice. Analogical learning generally involves developing a set of mappings between features of two instances and such a technique is often called case-based reasoning.

Proposal: the use of analogy as a tool for the description of situations limited by the assertion of a difference certain

In our example, the situation description is enriched by this analogical reasoning technique, associating the services listening with a phenomenon similar but less complex: the acoustic mode (line 29). DOLCE-Lite and the D&S plugin lacking of this very important property, in Soa2m we extended the foundational ontology by adding this concept as an extension of the taxonomy presented in [Bellik 1995]. This means that if more information is needed to understand the behavior of the services listener, a reasoner can use the information available concerning the acoustic mode while we express at least one difference certain. In result, this mechanism significantly reduces the annotation effort.

Situation semantics in Soa2m are intended to capture and represent human experience, and to describe on a high-level the occurrences in which humans participate. And this, under a variety of aspects such as time and space, objects and persons involved, as well as mereological, causal, correlative and inheritance relationships between situations.

With this goal, the **usage** ontology relates the current individual situation «myServicesListener» with other entities available in the Soa2m situation ontology:

it expresses that the situation has as participants, entities of type «listener» and «advertiser» (line 30 to 31) ,

it expresses that it is performed «duringCapture» in some «public» place (line 32 to 33)

and finally, it expresses that the thing that is listened is an «announcement». Finally, the ontology informs that any entity named with any of the listed tags in lines 35 to 43 can be considered as equivalent to a services listener.

```
30        <Soa2mSituation:hasWhoActor  rdf:resource="&listener;listener"/>
31        <Soa2mSituation:hasWhoOriginator  rdf:resource="&listener;advertiser"/>
32        <Soa2mSituation:hasWhen   rdf:resource="&mode;duringCapture"/>
33        <Soa2mSituation:hasWhere   rdf:resource="&mode;public"/>
34        <Soa2mSituation:hasWhat  rdf:resource="&listener;announcement"/>
35        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag> ADVERTISEMENTS_LISTENER </xm:Tag></rdfs:label>
36        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag> ADVERTISEMENT_LISTENER </xm:Tag></rdfs:label>
37        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag>  CONTROL_POINT </xm:Tag></rdfs:label>
38        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag> SERVICES_LISTENER </xm:Tag></rdfs:label>
39        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag>  SERVICES_MONITOR </xm:Tag></rdfs:label>
40        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag> SERVICES_OBSERVER </xm:Tag></rdfs:label>
41        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag> SERVICE_LISTENER </xm:Tag></rdfs:label>
42        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag> SERVICE_MONITOR  </xm:Tag></rdfs:label>
43        <rdfs:label rdf:datatype="&rdf;XMLLiteral"><xm:Tag>  SERVICE_OBSERVER </xm:Tag></rdfs:label>
44      </owl:NamedIndividual>
45      <owl:NamedIndividual  rdf:about="&listener;listener">
46        <rdf:type rdf:resource="&who;actorValue"/>
47        <mode:analogically-references   rdf:resource="&mode;receptor"/>
48      </owl:NamedIndividual>
49      <owl:NamedIndividual  rdf:about="&listener;advertiser">
50        <rdf:type  rdf:resource="&who;originatorValue"/>
51         <mode:analogically-references   rdf:resource="&mode;emitter"/>
52      </owl:NamedIndividual>
53      <owl:NamedIndividual  rdf:about="&mode;duringCapture">
54        <Soa2mSituation:temporally-starts  rdf:resource="&listener;discovery"/>
55      </owl:NamedIndividual>
56      <owl:NamedIndividual  rdf:about="&listener;discovery">
57        <rdf:type  rdf:resource="&which;whenValue"/>
58      </owl:NamedIndividual>
59      <owl:NamedIndividual  rdf:about="&listener;announcement">
60        <rdf:type  rdf:resource="&ExtendedDnS;information-encoding-system"/>
61        <mode:analogically-referenced-by  rdf:resource="&listener;service"/>
62      </owl:NamedIndividual>
63      <owl:NamedIndividual  rdf:about="&listener;service">
64        <rdf:type  rdf:resource="&ExtendedDnS;information-encoding-system"/>
65      </owl:NamedIndividual>

66      <owl:ObjectProperty  rdf:about="&Soa2mSituation;temporally-starts">
67        <rdfs:subPropertyOf  rdf:resource="&DOLCE-Lite;mediated-relation"/>
68        <rdfs:domain  rdf:resource="&DOLCE-Lite;particular"/>
69        <rdfs:range  rdf:resource="&DOLCE-Lite;temporal-region"/>
70        <owl:inverseOf   rdf:resource="&Soa2mSituation;temporally-starts"/>
71      </owl:ObjectProperty>
```

In the example, the ontology also affirms that «duringCapture» in the acoustic mode is a property shared with the services listener, and this capture «temporally-starts» in a semi-interval of time[143] beginning with the achievement (a perdurant) called «discovery» and without any planned end (line 53 to 55 and line 66 to 71).

[143] Semi-intervals are described in supra section 1.2.3.1 Temporal Situation and are a contribution of this thesis to the DOLCE-Lite ontology.

Proposal: Two types of ontologies for a semantic method of annotation

To sum up, the Soa2m project proposes a method for the annotation of processes, represented by the Modality Component Services. This method consists on the combined use of functional descriptions of the service with **service** ontologies and non-functional descriptions with **usage** ontologies. Both kind of ontologies are linked to a WSDL document and more precisely, to some elements of the service description in WSDL that are candidates to a semantic extension of their mean.

These ontologies allows the use of more expressive filters in the selection of modalities in the *fusion* and *fission* processes. These filters can be based on context and usage information provided in the situation description. For example, the selection can be based on the hasWhere property expressing that the services are designed to work in public spaces (line 33) or the services using media only of a specific mode.

The extensions for the WSDL elements can be categorized on four types focused in the enhancement of the discovery process:

- extensions oriented to help in the **intention discovery** with **usage** descriptions of the service and interfaces
- extensions oriented to support the **semantic discovery** with **usage** descriptions of the WSDL operations, request data and response data.
- extensions oriented to support the **behavior discovery** with **service** descriptions of the WSDL operations, request data and response data.
- extensions oriented to support the **capacities discovery**, with **service** and **usage** descriptions of the WSDL dataTypes.

Finally, usage ontologies are situation structures that follows a data model provided by Soa2m as an ontology library that will be described in the next section.

2.2.3 Semantic Description of Multimodal Data

In the field of research of web services, as in multimodal systems, there is still no standardized solution addressing what service providers (or Modality Component providers) should expose or advertise in service descriptions as non-functional properties.

Non-functional properties in services

For the web services field, some catalogs are proposed [Becha et al. 2012], covering more or less an initial set of the more common non-functional properties (cost, completion time, trust, availability, reliability, usability, validity, standards compliance, dependencies, failure mode, security, execution models, jurisdiction, life-cycle updates, penalty rate, compensation rate, resource utilization, throughput, and accessibility). However, these are mainly properties concerning the contract agreement.

The Webintents proposal

In a less expressive but a lighter approach, the Web Intents framework [W3C-WEBINTENTS 2012] proposes the advertisement of an intention in which the service announces to be able to handle an action on the user's behalf and the *media* supported.

As a consequence of its genericity, the Intent is a kind of non-functional property of the service. Applications request to start an Action of a certain verb (share, edit, view, pick etc.) and the system will find the appropriate services to use based on the user's preference. These actions are supposed to be well known, but for the moment, any formal catalog of intents is proposed.

Requirements for the advertisement of multimodal properties

A similar effort is needed for the MMI Framework & Architecture recommendation, because the description of the non-functional properties of Modality Component services is indispensable for the management of the Modality Component life cycle - discovery, register, composition and distribution-.

Moreover, in *multimodal systems*, another kind of non-functional properties are needed to handle the context-awareness in the multimodal interaction cycle, by adapting the multimodal composition to a particular kind of situation or space.

These properties should describe environmental attributes that may affect or enhance the multimodal interaction.

These are annotations that go beyond the functional properties or the service agreement and the properties concerning the Quality of Service (QoS), to focus in the context of execution related to the multimodal interaction domain.

To handle this kind of information, a high level approach that aims to describe some static and dynamic stereotypes of real world situations is needed.

Outline

In the Soa2m architecture we propose a data model evolving from a controlled vocabulary to a taxonomy to finish with a knowledge model for situations. It evolves according with the complexity of the data to describe and the goals of the annotation, previously presented in section 2.2.2. The Semantic Annotation of Services.

The model is arranged in three optional but complementary parts: a basic glossary for discovery, a taxonomy for multimodal discovery and a knowledge model for multimodal discovery.

2.2.3.1 A Glossary for Multimodal Discovery

The Tagging process

The basic principle of tagging is that end users do subject indexing instead of experts only. This is a mode of description of entities based on collaborative efforts. It involves two steps: a conceptual analysis leading to a data model and a denotation procedure.

Conceptual analysis involves deciding on what a resource is about and what is relevant in particular. Note that the result of conceptual analysis heavily depends on the needs and interests of the domain that a resource is tagged for.

Denotation is the process of finding an appropriate set of index terms that represent a particular facet of an entity or process.

Two difficulties arise in the tagging process: the pertinence for the domain of this intuitive (based on common sense) data model; and the polysemic nature of language (the multiple number of meanings, interpretations or understandings of words).

Synonyms and homonyms/homographs are frequent problems in the process of denotation that a controlled Glossary tries to eliminate by providing a list of preferred and non-preferred terms, often together with definitions and a semantic structure.

This glossary can be defined by three means: a term lists like authority files in dictionaries that emphasize in entity definitions.

Classifications and categories also known as taxonomies that emphasize on the creation of subject sets.

And relationships groups like thesauri, semantic networks, and ontologies that emphasize the connections between concepts.

The multimodal taxonomies used in Soa2m

In the multimodal research field, glossaries are provided as taxonomies to model device modalities [Foley et al. 1984] [Buxton 1986] [Mackinlay et al. 1990] [Jacob et al. 1992] or representational modalities and media [Arens 1993] [Bernsen 1993].
On the other hand, the Extensible Multimodal Annotation Markup Language (EMMA) [W3C-EMMA 2009] is based on an annotation glossary for the media engaged in multimodal interaction, mostly for Modality Components providing recognition.

It provides a basic glossary but it is mostly oriented to annotate recognition processes and results.

Nevertheless none of these approaches is designed to support the annotation or tagging of bidirectional (in/out) multimodal entities as services, neither to enhance the Modality Component discovery and registering in a multimodal system. In other words, there is no explicit link between these theoretical and language proposals and a concrete and common mechanism for the description of multimodal components and its processes. This work remains to be done in a standardized way.

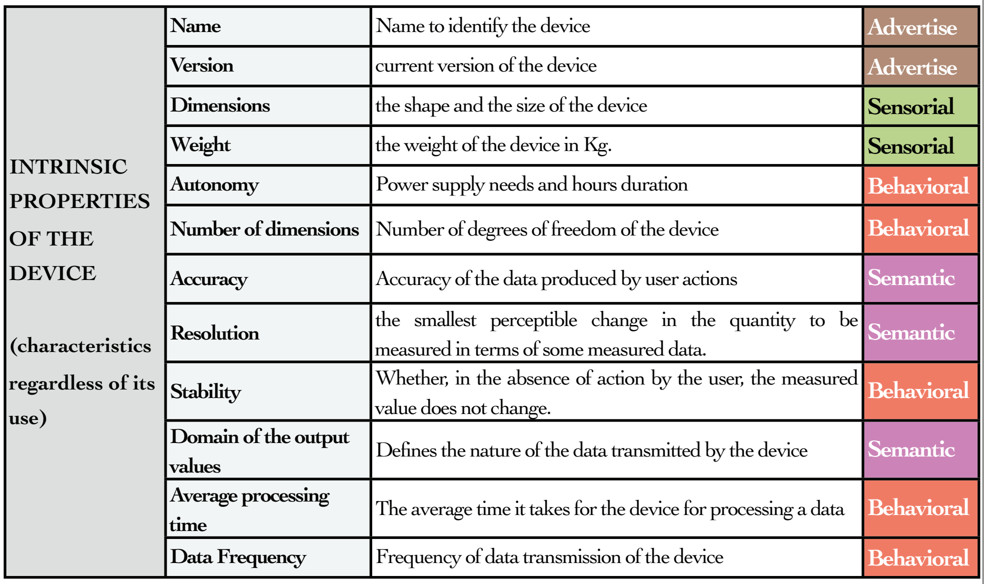
In Soa2m the focus is the dynamic discovery of Modality Component services using generic information about the underlaid processes.

This information is provided by an announcement and a service description advertised in some network.

With this goal on mind, we analyzed some current proposals of data models for the interfaces of multimodal components as starting points for the construction of a glossary.

For example, [Bouchet 2006] proposes a not-exhaustive list of properties[144] to describe input devices divided into three groups. The first group corresponds to the «intrinsic» properties [**Table 2.2.1**].

As we can in the last column of the table[145], this first group associates very divers kinds of data. For example, the name and version of a device coming from a manufacturer is a data depending on the product advertisement of manufacturers and mostly of the time, it does not carries semantics about the characteristics of the device.



INTRINSIC PROPERTIES OF THE DEVICE (characteristics regardless of its use)	**Name**	Name to identify the device	Advertise
	Version	current version of the device	Advertise
	Dimensions	the shape and the size of the device	**Sensorial**
	Weight	the weight of the device in Kg.	**Sensorial**
	Autonomy	Power supply needs and hours duration	Behavioral
	Number of dimensions	Number of degrees of freedom of the device	Behavioral
	Accuracy	Accuracy of the data produced by user actions	Semantic
	Resolution	the smallest perceptible change in the quantity to be measured in terms of some measured data.	Semantic
	Stability	Whether, in the absence of action by the user, the measured value does not change.	Behavioral
	Domain of the output values	Defines the nature of the data transmitted by the device	Semantic
	Average processing time	The average time it takes for the device for processing a data	Behavioral
	Data Frequency	Frequency of data transmission of the device	Behavioral

Table 2.2.1: Intrinsic Properties of Input Devices from [Bouchet 2006]

In contrast, dimension and weight are very specific data that concerns the sensorial level, as it can be validated in an empiric way. Accuracy and resolution depend on «abstract regions»[146], these are measures coming from social agreements in an specific domain.

And finally, all the other properties, concerns the behavior of the device, which is naturally related to its use. In result, they cannot be viewed as characteristics «regardless of the devices's use».

144 This list is very close from the properties given in [Arens et al. 1993]. In both cases our comments are the same, so we decided to present only the more recent proposal.

145 The labels and colors of the last column corresponds to the conceptual model defined in the Soa2m project and presented in section 2.1.1.2. Abstract Layers of the Architecture.

146 In DOLCE terms, an abstract region is a conventional (social) value given to a certain measure.

Then, one of the difficulties raised by this proposal it its granularity. If this glossary is used as a data model for the description of a Modality Component service, the information provided is too fine-grained to produce an accurate filter for an initial gross selection or to begin a discovery process.

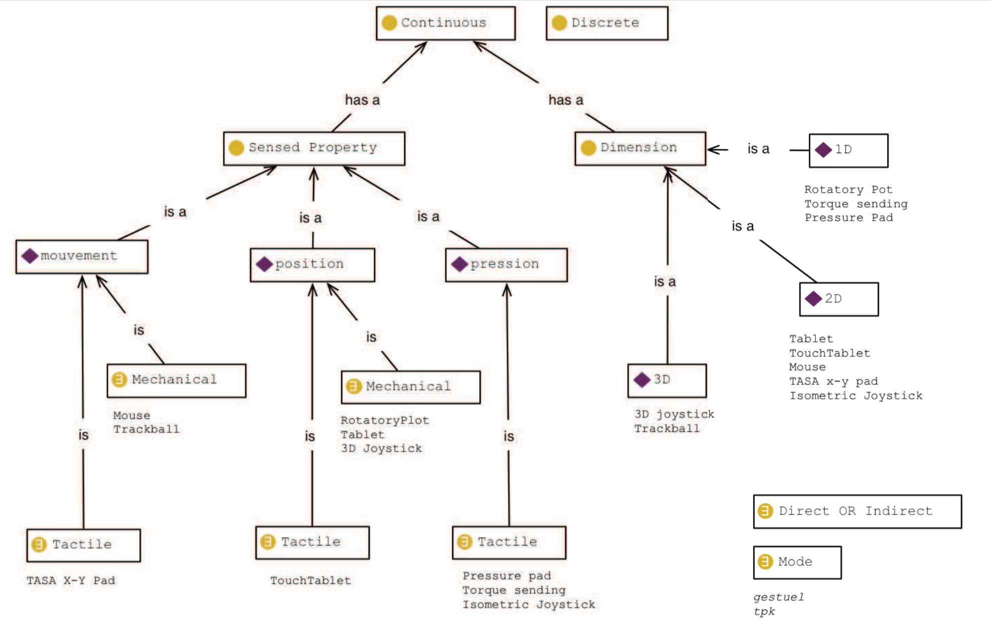


Figure 2.2.12: Taxonomy of Buxton for hand-controlled input devices

The same difficulty is present on the proposal of [Serrano 2010], where, based on the Buxton taxonomy [Figure 2.2.12], it defines the interfaces of modality components from a Device perspective [Table 2.2.2] in a fine grained perspective.

DEVICE INTERFACES	Property	Movement	Behavioral
		Pressure	Behavioral
		Position	Sensorial
	Number of dimensions	1D	Semantic
		2D	Semantic
		3D	Semantic
	Data Frequency	Frequency of data transmission of the device	Behavioral

Table 2.2.2: Devices Interfaces from [Serrano 2010]

To overcome this difficulty, we decide to analyze a higher level of description included in the same proposals: the other groups of properties listed by Bouchet and the Task component interfaces proposed by Serrano.

[Bouchet 2006] proposes a not-exhaustive list corresponding to the properties used to deploy devices intended to be adapted to the human use and the interaction situation, and properties to reflect the component's state [Table 2.2.3]:

PROPERTIES RELATED TO THE HUMAN USE	**Portability**	Specifies whether the device is portable by the user or not	Sensorial
	Level of expertise required	the difficulty for a user to use the device	Semantic
	Communication mode	Refers to organs used when a user communicates with the computer system	Sensorial
	Nominal place of interaction	Where the user must focus its attention in order to provide data to the system (proxemics)	Semantic
PROPERTIES AT RUNTIME	**Running State**	The current state of the device	Sensorial
	Failure State	Malfunction of the device	Semantic
	Data Trust factor	Confidence factor defining the relevance and credibility of the information generated	Advertise

Table 2.2.3: Other Properties of Input Devices from [Bouchet 2006]

As we can see, different conceptual levels are mixed again, but some generic information that can be used in a description. For example, some contextual information in the nominal place of interaction or some perception situation data in the communication mode.

This property, the communication mode, is inspired from the controlled vocabulary of modes presented in [Bellik 1995] with a set of criteria provided to classify modalities:

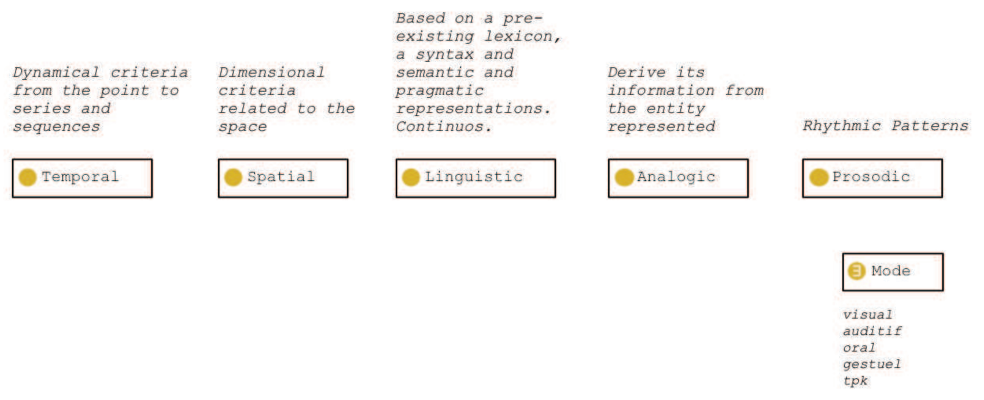

Dynamical criteria from the point to series and sequences
Dimensional criteria related to the space
Based on a pre-existing lexicon, a syntax and semantic and pragmatic representations. Continuos.
Derive its information from the entity represented
Rhythmic Patterns
Temporal
Spatial
Linguistic
Analogic
Prosodic
Mode
visual
auditif
oral
gestuel
tpk

Figure 2.2.13: Criteria for a Taxonomy of Modalities from Bellik

Unfortunately, in the properties list, only the classification of modes is borrowed from these high level criteria that is an interesting starting point to propose a data model for the annotation of Modality Components, for example, with the proposal of the analogy as a criteria of classification [**Figure 2.2.13**].

On the other hand, [Serrano 2010] proposes a special category of components, the Task Components.

These component are highly reusable, independent and their interfaces are designed based on the taxonomy of Foley [**Figure 2.2.14**] which is oriented to separate the interaction tasks from the concrete implementation of the device.

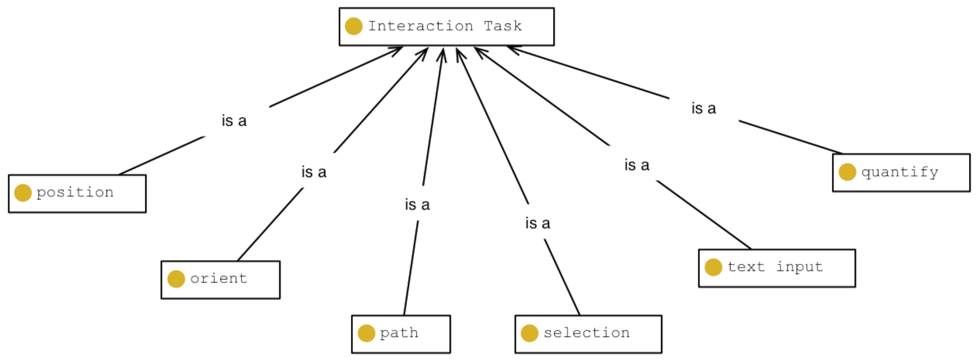


Figure 2.2.14: Taxonomy of Foley et al. for input devices

The taxonomy is structured around the graphics subtasks that input devices can perform in a very abstract-oriented way. This approach, similar but more complete than the Web Intent proposal, is based on a theoretical analysis of the more relevant tasks available on the input devices at the time, and completed with classification examples (not included in [**Figure 2.2.14**]). In this sense, Foley's et al. taxonomy was a complete glossary of abstract interaction task at the time, and it keeps its validity today thanks to its generic approach.

So, the task component interface proposed in [Serrano 2010] uses the taxonomy of Foley as values for its classification [**Table 2.2.4**]:

TASK COMPONENT INTERFACES	Type	Selection	Semantic
		Position	Semantic
		Orientation	Semantic
		Path	Semantic
		Quantification	Semantic
		Text Input	Semantic
	Number of dimensions	1D	Semantic
		2D	Semantic
		3D..nD	Semantic
	Domain	Domain of the dimension	Semantic

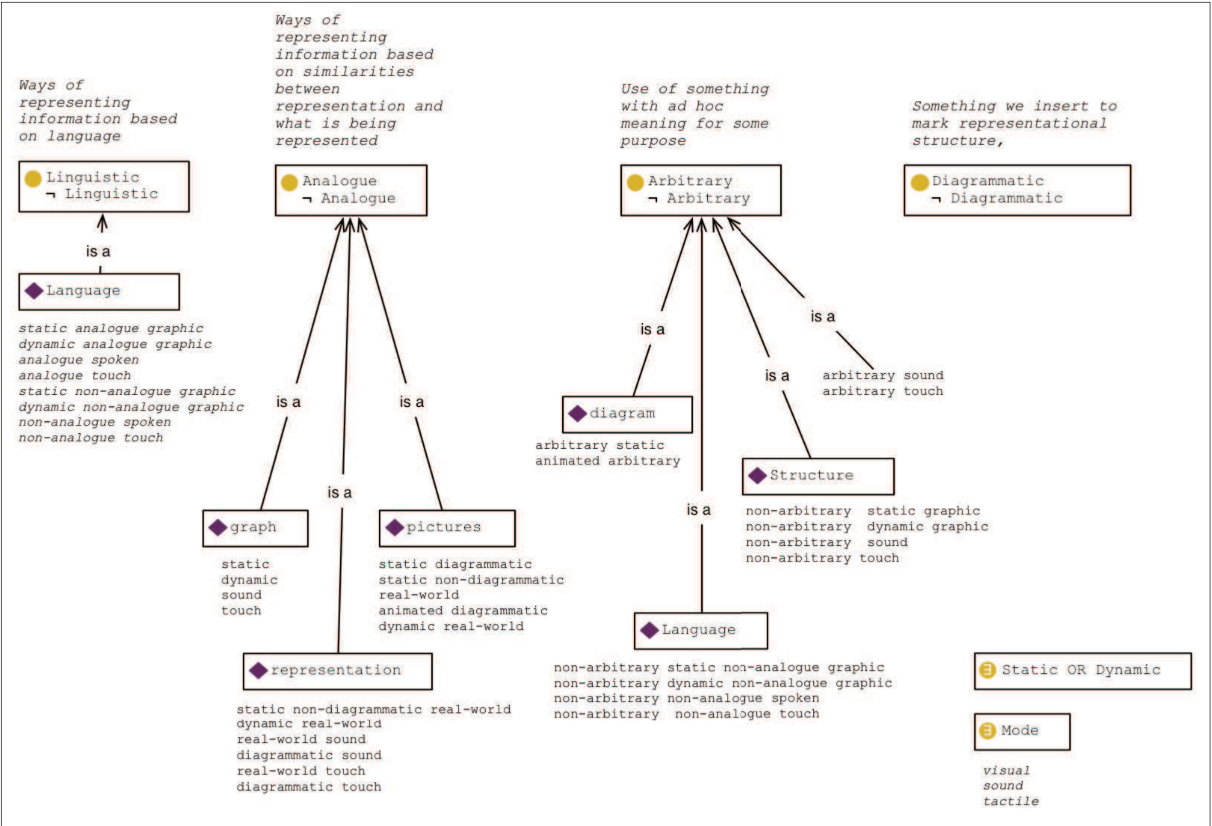
Table 2.2.4: Task Component Interfaces from [Serrano 2010]

In consequence, from this description we can highlight the proposal of types that can be used for the classification of Modality Component services while the information about the number of dimensions is still too fine grained to be used in a process of services discovery.

Finally, we completed our analysis with a classification provided by a generic taxonomy of modalities proposed by [Bernsen 1993] and designed for unimodal representational modalities (input and output).

An unimodal modality is an un-composed way of representing information. Representations that are composed of two or more unimodal representation forms are called multimodal. For example, presentations are multimodal as they are composed of a sequence of graphics with textual annotations. The graph and the annotations are both unimodal.

The taxonomy of Bernsen [**Figure 2.2.15**] is based on the state as a classification point of view: static/dynamic; and a set of four criteria that are binary opposites: linguistic/ non-linguistic, analogue/non-analogue, arbitrary/non-arbitrary, diagrammatic/non-diagrammatic. And finally all the data is expressed according to a *mode*: visual, sound or tactile.



Ways of representing information based on language
Linguistic ¬ Linguistic
is a
Language
static analogue graphic
dynamic analogue graphic
analogue spoken
analogue touch
static non-analogue graphic
dynamic non-analogue graphic
non-analogue spoken
non-analogue touch
Ways of representing information based on similarities between representation and what is being represented
Analogue ¬ Analogue
is a
graph
static
dynamic
sound
touch
pictures
static diagrammatic
static non-diagrammatic
real-world
animated diagrammatic
dynamic real-world
representation
static non-diagrammatic real-world
dynamic real-world
real-world sound
diagrammatic sound
real-world touch
diagrammatic touch
Use of something with ad hoc meaning for some purpose
Arbitrary ¬ Arbitrary
diagram
arbitrary static
animated arbitrary
arbitrary sound
arbitrary touch
Structure
non-arbitrary static graphic
non-arbitrary dynamic graphic
non-arbitrary sound
non-arbitrary touch
Language
non-arbitrary static non-analogue graphic
non-arbitrary dynamic non-analogue graphic
non-arbitrary non-analogue spoken
non-arbitrary non-analogue touch
Something we insert to mark representational structure,
Diagrammatic ¬ Diagrammatic
Static OR Dynamic
Mode
visual
sound
tactile

Figure 2.2.15: The Taxonomy of Modalities from [Bernsen 1993]

The Bernsen taxonomy can be aligned with DOLCE

If we take a close look to this taxonomy, we can see that it is possible to map the point of view based on static or dynamic data to the classification concepts proposed by DOLCE-Lite: endurants and perdurants. We also see that the mode is a basic criteria of disambiguation of modalities, and can be take into account as [Bellik 1995] and [Bouchet 2006] already showed. Finally, we can suppose that it is possible to extend the classification of output modalities to a more complete classification including input modalities.

In conclusion, the taxonomy of Bernsen is an important basis to explain the meaning of the annotation terms for a controlled vocabulary in the form of a Glossary for multimodal systems.

Proposal: A Glossary for discovery & Registration divided on two parts

In Soa2m this Glossary is proposed to enhance the discovery of multimodal services and is related to the terms presented in the sections 1.1. Multimodal Terminology used in this Research and 1.2 Definition of a Multimodal System.

The Soa2m glossary is divided in two parts.

- **The first part** concerns the subsumption[147] relations [**Table 2.2.5**]. It is structured on tags classifying the Modality Component services according to their membership or association to a class. In this way, *modality* and *media* are described in conformance to the *modes* handled by the Modality Component services. This first description allows a discovery by **intention** based on generic categories concerning directly the multimodal domain.

147 Subsumption is to place a concept under another as belonging to it; to include or contain under something else. To consider an occurrence as part of a principle or rule.

TERMS BASED ON SUBSUMPTION	**Name**	The intentional name of the service	Semantic
	Affiliation	The high level service category to adhere (to parse using generic tasks described by a multimodal taxonomy or ontology)	Advertise
	Modality	A list of the supported modalities classed by mode (to parse using generic modalities described by a multimodal taxonomy or ontology)	Semantic
	Medium	A list of the supported media classed by mode (to parse using generic media described by a multimodal taxonomy or ontology)	Semantic

Table 2.2.5: Glossary Terms based on subsumption relations

The first term, «Name» corresponds to the intentional scope name described in section 2.2.1.3 Registration, and is used to announce the service in the network. It is a compound name in three parts and provides the semantics [**Figure 2.2.16**] about the implementation of the component, its most important attribute and its more important role:

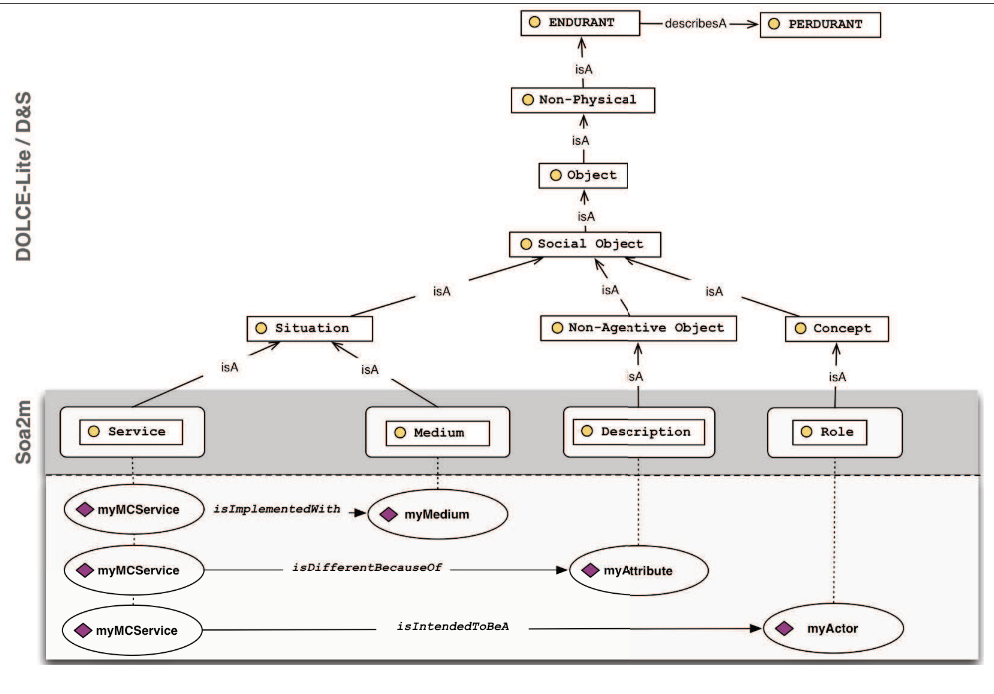


Figure 2.2.16: Definition of the intentional scope name in MC services

Based on this term, a Soa2m Controller Component can parse the information and classify the service in a high level perspective. The triplet is inspired on the intentional name schema [Adjie-Winoto et al. 1999] and show hierarchical tree relationships between general concepts (including some negative aspects). These names are intentional; they describe the service intent in the form of properties and attributes.

The name resolution in the Controller Component is supported by the use of the Soa2m taxonomies and ontologies, or by the use of a restricted vocabulary hard-coded in the multimodal application.

The second term «Affiliation» specifies the generic information about some complementary association of high level. It puts forward an aspect of the service that could affect its selection and is part of the advertisement policy of the service. It can be for example, a complementary inheritance relationship or a relation based on some association or analogy [**Figure 2.2.17**].

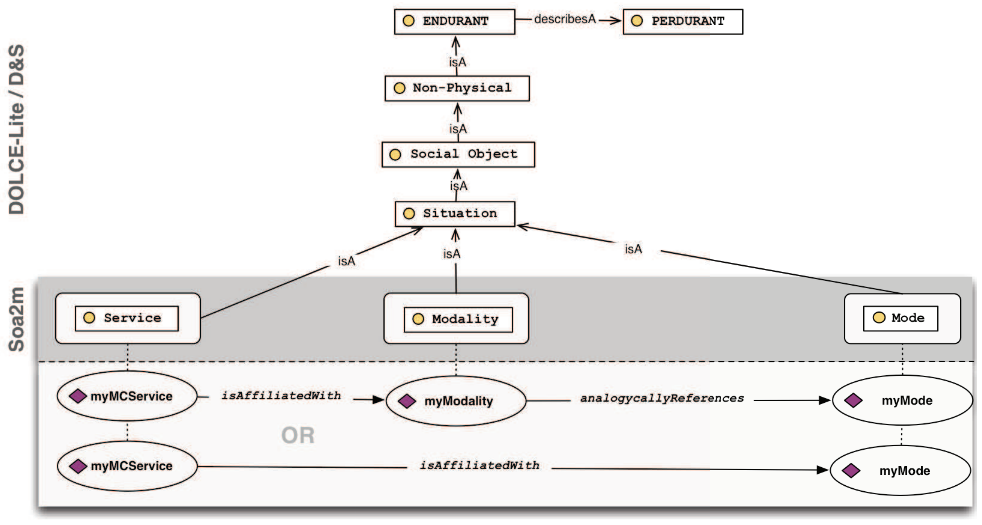


Figure 2.2.17: Definition of the Affiliation in MC services

In the first case, the affiliation can be related to a particular «myModality» that is in some extent associated to the particular «myMCService». In the second case, the affiliation can be asserted about other particular which is related with the service by an analogical relationship.

For example, a myVoice Synthesizer service is affiliated with myVoiceModality by association, but it can be also affiliated with myAcousticMode by analogy (in the emission side).

The third term «Modality»[148] specifies the form in which some information is realized according to a precise *mode*. A *modality* is described by a situation and contains a recognizable logical structure: the information object. For example, gesture *modality* is the realization of some intention in a situation of communication (the information object is the «message» to communicate). Thus, to draw a cross with a hand realizes the benediction message.

In [**Figure 2.2.18**] «myMCService» actsIn some *mode* that is perceived by a final user through a *modality* that is part of a *medium*, e.g. a face synthesis service actsIn the visual *mode* that is perceived through a 3D mesh *modality* that is part of an avatar medium.

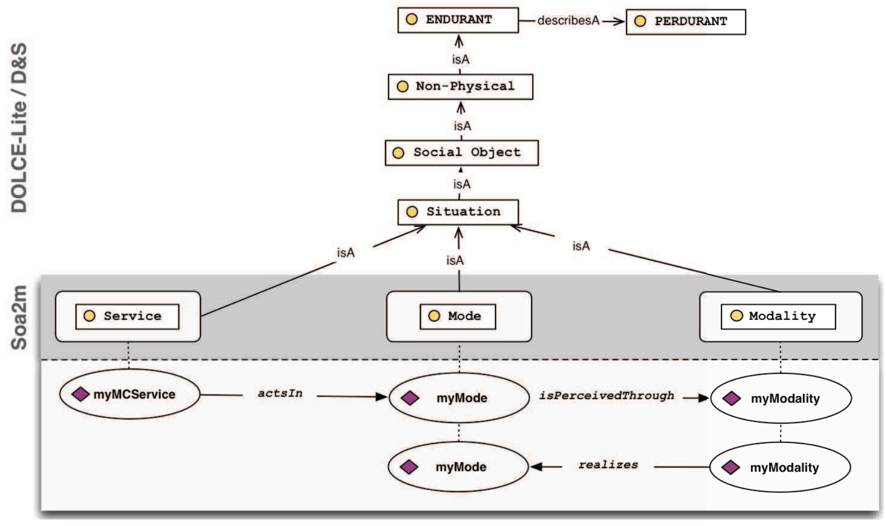


Figure 2.2.18: Definition of the Modalities in MC services

148 See supra section 1.1.4 Modality.

Then, the last term of this subsumption part is the «Medium».[149] In Soa2m a *medium* is a technical entity supporting a limited number of modalities according to the semantics of the message and the capabilities of the support itself. A MC service acts with *media*. Each *medium* uses one or more *modalities* that realizes some *mode*. [**Figure 2.2.19**] This term defines the list of *media* participating in the service, ordered by importance and by mode. For example, a gesture recognizer service actsWith the sign language *media*, using the single hand gesture *modality* that realizes the haptic Mode and is perceived in the visual mode.

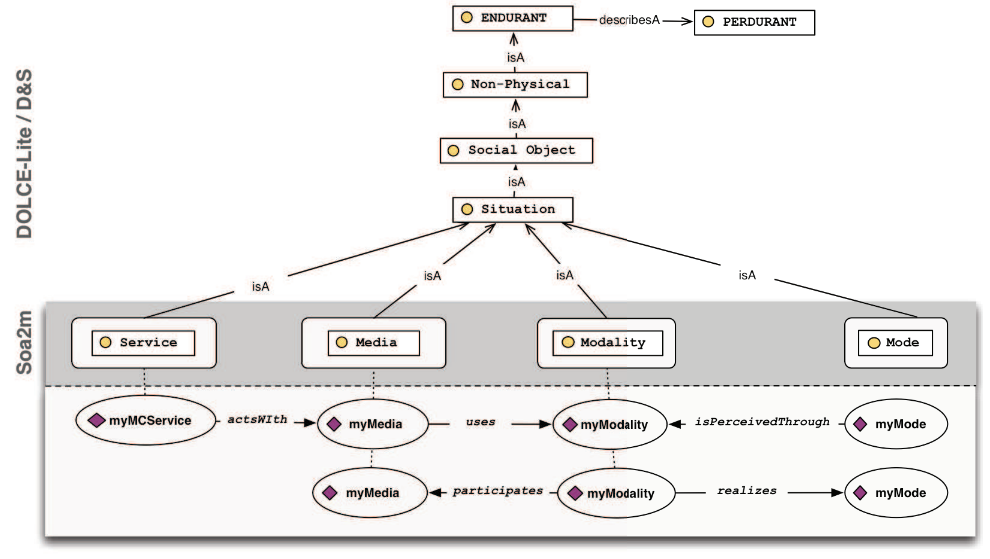


Figure 2.2.19: Definition of the Medium in MC services

To sum up, the first part of the data model concerns the subsumption relations that will allow a gross selection of Modality Component services based on intentionality. This part responds to the working hypothesis of Soa2m «it could be possible to build a system bringing services to each user with the use of the semantically best resources available, independently of the communication channel, the mode of restitution, the media or the device» by proposing a first selection based on high level information about the behavior and some important specificity of each service.

In this way, for example, if the user intention is to raise an alert in a critical situation, the «Alerter» MC services will be selected, and by analogy also the «Notifier» MC services, in all the available modes, modalities and media. Starting from this reduced set of possibilities the *fussion* and *fission* processes will be capable to execute the final instantiation of services with all the complementary details associated to the current situation of interaction.

The second part concerns the life-cycle and the general behavior attributes from a qualitative point of view [**Table 2.2.6**]. This part is structured on five attributes described by triplets of values.

The «Life» term [**Table 2.2.6**] corresponds to the registering of the service to be part of the multimodal system. It is inspired from the ServiceData attributes of the Open Grid Services Architecture [Foster et al. 2002] and represents a) the temporal moment when the service is registered: «GoodFrom» b) the final moment of validity for this registration opening a period to allow a new registering and the update of the service metadata: «GoodUntil» c) the final moment when registration is no more allowed «NoGoodUntil». To include this term in a description will allow to handle the state of the system and to renew the information about the available services.

The «State» term [**Table 2.2.6**] corresponds to a triplet representing a) the current condition of the service for example: «Waiting» , «Looping» , «Loading», «Sleeping» b) the temporal evolution of the state, for example: a timestamp or a symbolic description «Currently», «Until», «Since».) the directional aspects of the state in symbolic terms. (what is the dynamics of the state of the entity?)

149 See supra section 1.1.5 Medium

For example: «Repelled», «Attracted», «Immobilized». As a result, a MC service in a deadlock state can be annotated with the triplet <SLEEPING, SINCE, PARALIZED> that informs that the component is in the sleeping state, it does not have any information about when this sleeping state will end[150] (for example in the case of a failure in the recovering of the session data from the Controller Component) and for this reason it is immobilized.

TERMS BASED ON LIFE-CYCLE AND BEHAVIOR	Life	GoodFrom	Sensorial
		GoodUntil	Sensorial
		NoGoodUntil	Sensorial
	State	Condition	Semantic
		Temporal Evolution	Semantic
		Directional Evolution	Semantic
	Type	Temporal Aspect	Behavioral
		Decisional Aspect	Behavioral
		Relational Aspect	Behavioral
	Skill	Talent	Advertise
		Weakness	Advertise
		Dexterity	Advertise
	Privacy	Data Policy	Behavioral
		Behavior Policy	Behavioral
		Inference Policy	Behavioral

Table 2.2.6: Glossary Terms based on life-cycle and behavior attributes

The «Type» term corresponds to the variations of the Soa2m architecture defined by the design patterns presented in section 2.1.1.4. Design Patterns of the Architecture.

The triplet represents:

a) the temporal support provided by the wrapped modality component in terms of two design patterns: the «forgetter» and the «chronicler». In the first case the Modality Component service does not have an explicit management of its current or past states because it does not dispose of a state manager. In the second case, the Modality Component service can keep track of the state of the interaction cycle, the participants or the environment and context because it has a state manager implemented.

b) the decisional support provided by the modality component in terms of two other design patterns: the «dumb» and the «smart». In the first case, the Modality Component wrapped by the service can't collect dynamic data or make decisions. In the second case, the Modality Component service represents an implementation that can make decisions based on some knowledge and is autonomous in some extent.

c) the relational aspect of the implementation according with two design patterns: the «outgoing» and the «shy» service type. The first can advertise its behavior and interfaces and communicate externally while the second does not provide a detailed description of its behavior and interfaces.

[150] In Soa2m the term «since» corresponds to the semi-interval of time in which we known the beginning point but the end point is unknown. The inverse interval is annotated with the term Until, which means that we know the final point even if we ignore when was the beginning.

As a result, a MC service can be annotated with the triplet <FORGETTER, DUMB, OUTGOING> that informs that the parser will found a detailed description of the service, that the service does not keep trace of its state and that it can not be autonomous. This is the case of a «Commander» MC service where the controls does not provide a pause and resume button neither visually adapt the interface to the context of interaction (by changing the color of buttons or providing data about the evolution of the process).

The «Skill» term represents the information that the designer of the modality component want to put forward to help in the selection of the service. It is an advertisement information related to the impression that the provider would like to convey. As a service provider it is always important to promote what we can do, what we are not capable to do and what we do best. This is the skill semantics. a) The talent is the more important aptitude of the MC service that the provider will emphasize. b) The weakness is the more evident disadvantage of the MC service that the provider would like to prevent. c) The dexterity is the proficiency in performing a given task.

For example, a MC providing Speech Recognition services, can be annotated with the triplet <SPEAKER_INDEPENDENT, NOISE, DATA_ENTRY>. The annotation expresses in the domain of speech recognition, that the principal talent of the service is to work without training data from the user, that it works very well in recognition of simple data entry like credit card numbers and that it works less well when there are noise in the space. Thus, the values of the annotations are domain dependent, but the semantics of the data model structure can be preserved and can be very useful for the selection of services, to refine the gross selection executed with the subsumption properties.

Finally, the «Privacy» term represents the ability to preserve some information about an individual or a group and reveal in each situation some information selectively. Then the privacy is related to the intention of contextualized anonymity.

In Soa2M a MC service can be selected according to its degree of privacy on three facets: a) the data policy reflects the level of anonymization in concrete information. This means that the MC service can ensure that personal data is not collected nor transmitted at one of this three inclusive levels: the identity, the relations and the categories. In the first case, information like a digital fingerprint must be anonymized, in the second case, information like what numbers I called by a voice commanded phone must be anonymized and in the third case, information about my gender coming from my voice pitch must be anonymized.

b) The behavior policy reflects the anonymization of the logs permitting to infer a given behavior. In identity anonymization the graph of the places I visit more frequently and the temporal patterns of my visits must be anonymized. In relation anonymization the graph of modalities I use more frequently must be anonymized and finally in categorial anonymization the inferences about my personality coming from my behavior must be anonymized.

c) The inference policy reflects the control over my social and semantic data at the higher level. In the first case, identity, my physical information must be anonymous, like my interaction handicaps inferred from my modalities choices. In the second case, my demographic information must be anonymous, like my age range inferred from some low level information, like my face recognition. In the third case, my categorial information must be anonymized, like my status of not-originally-national coming from my accent or my usage of multiple languages.

This last term will be more and more pertinent, as the semantic web of data and the internet of «intelligent» things will evolve. It is an important topic to reflect the reasoning or collection power of a Modality Component service but also its capability of adaptation and personalization.

Conclusion

In conclusion, in this section we introduced a limited Glossary proposed as a basic support for the discovery and registering of Modality Component services. This controlled vocabulary is the basis for two annotation mechanisms explained is section 2.2.2.1 Description of MC Services with Data Structures and section 2.2.2.2 Description of MC Services with RDF Manifests.

This Glossary contains nine terms divided on two groups and is the result of the analysis of two recent proposals coming from EMMA [W3C-EMMA 2009] and the ICARE and OpenInterface projects, three taxonomies of modalities (input and output) and the use of some approaches coming from distributed architectures.

The intent with this Glossary is to contribute as a starting point, to the discussion for an agreement in a common, generic and standardized vocabulary to ensure the discovery of modalities in large-scale networks: a process to be performed before any *fusion* or *fission* and depending on the state and architecture of the *multimodal system*.

The Glossary values are founded on a taxonomy of modalities created in the Soa2m project as a result of the same analysis, but out of the scope of this document. The reason for this choice is that we limited our presentation to the explanation of the data model that supports the semantic annotation mechanism.

However, it is possible to conclude from the Glossary that the data model behind important concepts like *mode*, *modality*, *medium* and interaction cycle is founded in the notion of situation coming from DOLCE-Lite. This notion is also at the heart of our semantic proposal and will be presented in the following section.

2.2.3.2 A Knowledge Model for Multimodal Discovery

In Soa2m the multimodal context is viewed as a situation that we can describe through the relationships between the participants under the umbrella of an intentional purpose in a given space and time.[151]

Proposal: the Soa2m situation

The Soa2m situation is a mental model, a part of the world, clearly recognized in common sense and in human language. The Soa2m situation is a tool to express the interaction phenomenon, based on previous knowledge of the given situational context: it is a multi-faceted characterization that describes the world of human experience in some aspect and with some granularity.

The Soa2m situation model consists of entities and the relations between those entities; or a set of relations between sets of properties. In result, this multi-faceted characterization is a formal representation that allows the capture and representation of occurrences in the real world.

Antecedents of the Soa2m situation

The starting point of the Soa2m situation model is the analysis of two data models from the perspective of the annotation of multimodal services to enhance discovery: the What Which How Then (WWHT) model presented in [Rousseau 2006] and the Event-Model-F presented in [Scherp et al. 2009]. This analysis is completed by the use of some methods for the description of situations coming from the multimodal performing arts and the multimodal film-making industry.

The **What Which How Then** (WWHT) model [Rousseau 2006] describes the «factual» aspects of the presentations (behaviours) characterized in terms of what information participates, which is the restitution of this information, how this restitution will be composed and when (then) it evolves.

This is a model for the composition of output multimodal presentations that proposes not only to take into account the presence of the parts of the presentation but also the way the presentation is composed. In other words, it covers the restitution performance[152] (instantiation) while preserving the importance of the selection of the presentation parts (allocation).

The model is based on four elements: the information, the interaction components, the behaviour and the interaction context.

Translated to a foundational ontology like DOLCE-Lite this elements and their relations reflect the mental model of the *state of affairs* of the multimodality according to the proposal [**Figure 2.2.20**].

151 See supra section 1.2.3 The Interaction Context of a Multimodal System

152 The performance in Soa2m and in the domain of the arts is the way of expressing things by an activity or task. The performance execution add some semantics to the message to execute, like in a song, which is interpreted (performed) in a way that extends the meaning of the music and the lyrics that are executed.

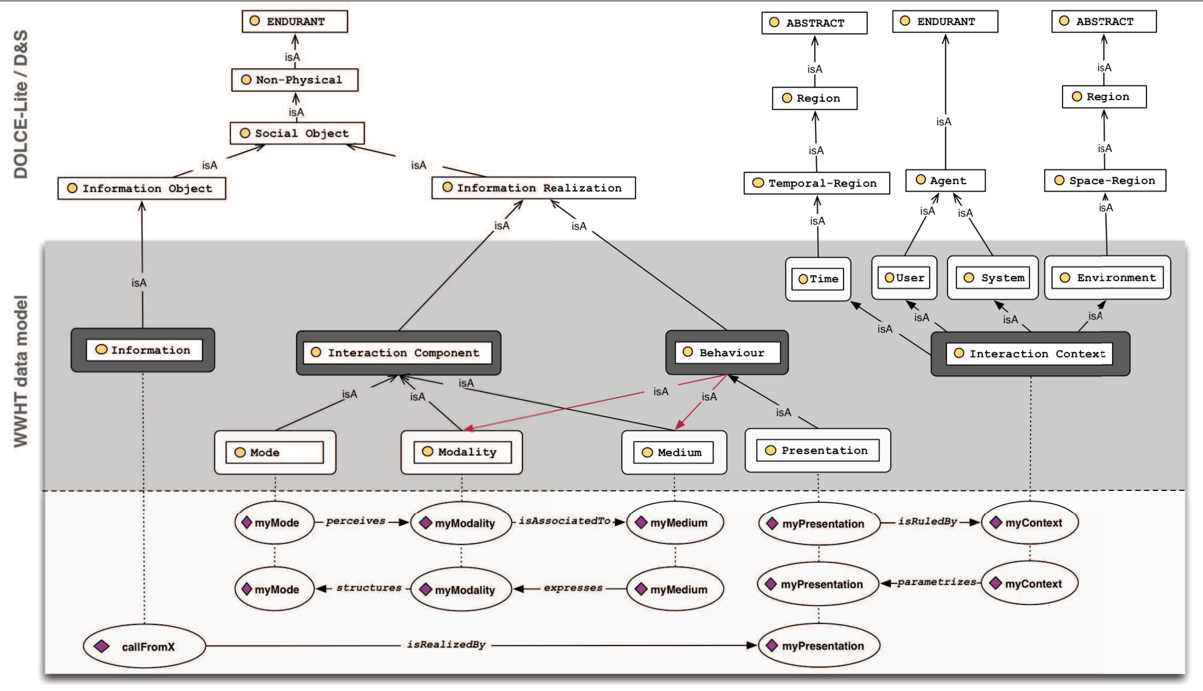


Figure 2.2.20: Basic Elements of the WWHT model aligned with DOLCE-Lite/DnS

We can follow the rationale of the model from the bottom of the figure to the top [**Figure 2.2.20**].

First, an information CallFromX (some message) is realized (presented) by a multimodal presentation.

The presentation is a «behaviour», composed of pairs of modality and medium and performance properties. This presentation, is parametrized with some rules created according to a description of an interaction context.

The context is a composite entity covering the description of time, the user, the system and the environment as external entities. Thus, the external interaction context affects the presentation instantiation and the allocation of the interaction components.

In DOLCE terms, this model « *reproduces modal reasoning into a first-order language adding time and world (or situation) parameters to the predicates* ».[153] This is an important difference with our model that is based on the premise that the situation not a parameter but the explication frame and the main criteria for the description of a multimodal interaction and all its participants. Our perspective in this aspect is a multiplicative approach, while the approach of the WWHT model is in some extent reductionist.

For this reason, and because the model is proposed mostly for the composition (*semantic fission*) of presentations and not for the intelligent discovery of modalities or media, the Soa2m situation model does not follow neither extend this approach.

Nevertheless, the WWHT model largely contributes to the enrichment of the Soa2m situation model:

- First, with its definition of the mode as a notion based on the human perception
- and second, with the interpretation of the multimodal interaction as a performed «behaviour».

153 See the quick introduction of DOLCE in Soa2m in supra section 2.2.2.3 Services Description with WSDL Documents

- The **Event-Model-F** presented in [Scherp et al. 2009] is intended to provide comprehensive support to represent time and space, objects and persons, as well as mereological (relations between parts and wholes they form), causal, and correlative relationships between events. For the authors, an Event occur or happen and is considered a perduring entity that unfold over time.

In contrast with the WWHT model, the Event model follows a *multiplicative* perspective, and aligns all its concepts to a lighter version of DOLCE: the DOLCE-Ultralight (DUL) ontology.

The fundamental approach of this model affirms [**Figure 2.2.28**]: myEvent (a perdurant) is included in myEventSituation. This situation, is a view of a context, is created by an observer and satisfies some description (myEventDescription).

This description of the event (myEventDescription) defines among other classifiers[154], the event type (myDescribedEvent).

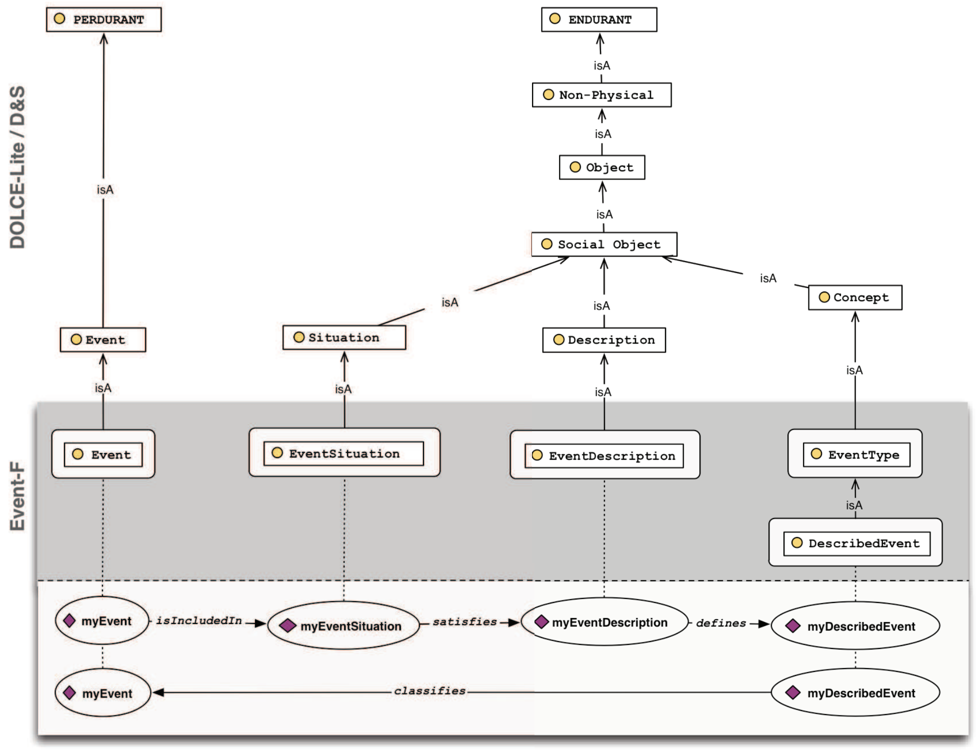


Figure 2.2.21: Basic Elements of the Event-Model-F aligned with DOLCE-Lite/DnS

In other words, a situation includes events; it is the point of view of an observer expressed in a description; and the description defines a series of classifiers for the events included in the situation. [**Figure 2.2.21**]

The Event-model-F is designed as a tool for describing real-world information in the domain of media-annotation. It is provided to document real-life occurrences, with multiple interpretations and points of view.

Whence its emphasis on the event description as an observer (interpreter) perspective of a concrete situation. It is also, for its documentary aspect, very oriented to the representation and knowledge of real-life.

In contrast, the Soa2m situation is less concentrated in this interpretation aspect to rather focus on the intention aspect. Its goal is also different. The Soa2m situation need to address the generation of situations more than their documentation under some interpretation of the experienced event.

[154] The classifiers proposed in the model correspond to the six design patterns: participation, mereology (to be part of), causality, correlation, documentation and interpretation. Our example in the figure is part of the pattern participation.

The description in the Soa2m situation must play the active role of a plan (to prepare in advance and design a multimodal experience), while in the Event-model-F it is an analysis (to comprehend and understand a real-world event).

Nevertheless, the instruments offered by this model are useful for our goal:

- First, the idea of a situation that is described by classifiers.
- Second, the active participation of an observer in this model can be used by analogy in the description of the role of an agent in the multimodal situation performance (instantiation).
- And finally, the Soa2m situation inherits from this model the use of the Description & Situations plugin of DOLCE to design and implement an ontology.

The **Script-Breakdown** perspective is the third element that contributed to the proposal of a Soa2m situation description.[155] A script breakdown is a detailed description of a complex situation (the plot) cut into a set of atomic situations. It is used by the production department on a set, basically to ensure the production includes all of the elements that are necessary to put together to create the final film and to allow the director to decide the more expressive way to combine this elements to carry a given message.

Thus, the script breakdown is a faceted description conforming to a plan, addressed to the different teams of the production crew under different granularities. This description is intended to guide a change of modality: the script is break down because a script is words and a movie is visuals. The breakdown is the first step from a cognitive *mode* toward a visual *mode* of storytelling.

The script breakdown goes beyond the simple shotlist which is a kind of checklist just to make sure that everything is covered. It is a tool to load semantically every element in the film to support the intended expression of the story. In other words, it is not only the ***What*** will be presented, but also it is closely related with ***How*** to present and ***Why*** we want to present the story this way.

Hence, the first part of the script breakdown is to start thinking about production design and the writer's and director's intent of communication with an observer. For the production crew, this is the point where they start figuring out the context of each situation and the context of the whole movie: what equipment and sets and wardrobe and special effects and visual effects and locations and makeup etc. they need in order to give to the movie a specific look and expression identity, which means it ties directly into the semantics of the film, but also in the production planning and budgeting.

As a result, in film-making, there is no movie without context. It does not exist an entity called «movie» to which we add location and temporal parameters to complete its semantics. The movie is a complex and recursive addition of semantically loaded contexts, evolving on time and having an identity that «happens» and «unfolds» during the experience of the film projection. In other words, a movie is a continuous entity: a perdurant.

Today, we can recognize that the film-making industry has achieved a great maturity in situation generation, based mostly in its technical approach in the process of description.

This is a method that evolves form an inherited craft with centuries of expertise in drama staging in the performing arts (dance, music, opera, theatre, magic, circus arts and musical theatre). In these arts, the pre-production method follows basically the same principles, while the resulting forms of expression are different.

155 This explanation is based in our personal ten years expertise of professional work in contemporary media arts.

One of the most interesting points for multimodality in computer sciences, is that since the beginning of the XX century, the contemporary performing arts have deeply explored the active participation of the public in the representation[156].

In multimodal words, the methods of production and description of situations in performing arts are oriented to the symmetric interaction, covering inputs, outputs and their feedback relationship. This means that more than a century of expertise is available to exploit processes and mechanisms of multimodal situation description and generation, in computer multimodal systems. For this reason, in Soa2m the situation data model for the Modality Component service description is inspired on this pre-production methods.

First, we recover the four facets of the description process. In theater, for example, the script is described to the pre-production crew at four inclusive levels:

- a description is produced for people in charge to find or create things, as a checklist: this is the **sensorial facet** that lists the participant things needed;
- a description is produced for people in charge to exchange and synchronize their exchanges, people that interact, like the cast actors, or the stage lighting design team and the scenery design team: this is the **behavioral facet** and it describes in timelines the behavior of persons and things;
- a description is produced for the assistants of the director and the team chiefs giving the general idea of the meaning of things and behaviors: this is the **semantic facet** and it describes the social meaning that will be communicated to the audience on each thing and behavior;
- and finally, a description is produced mostly by the director for the producer, the investors and the press team: this is the advertisement of the **intentional facet** that describes the personal or collective message that the piece will convey, its communicative goal and the mental model of the world it is intended to transmit.

Second, in Soa2m we recover the three granularities of the description process. In vocal music, for example, the song can be described in three complementary granularities:

- a description is produced to focus the generation and articulation of sounds and notes form the basis for the musical instructions, e.g. the exact execution of the partition. This is the **punctual granularity** that focus on the atomic components of a situation;
- a description is produced to focus in sounds emitted to express feelings . This can be produced during the performance by exaggerating the gestures in the performance or the rhythm[157], tempo[158], phrasing[159], dynamics[160] and pitch[161]. The use of these musical tools is described to indicate how to create an emotional relation between the audience and the sound, e.g. the description of the body expressions of a musician related to the partition. This is the **relational granularity** that focus in the means needed to relate things and the nature of their relations.

156 For example, a performance in contemporary art is a situation that involves four basic elements: time, space, the performer's body, or presence in a medium, its interpretation point of view and a relationship between the performer and the audience.

157 «movement marked by the regulated succession of strong and weak elements, or of opposite or different conditions» according to the New Oxford American Dictionary.

158 «the speed or pace of a given musical piece» according to the New Oxford American Dictionary.

159 « the manner of playing the individual notes of a particular group of consecutive notes and the way they are weighted and shaped relative to one another» according to the New Oxford American Dictionary.

160 «the volume of a sound or note, but can also refer to every aspect of the execution of a given piece, either stylistic (staccato, legato etc.) or functional (velocity).» according to Wikipedia.

161 «a standard degree of highness or lowness used in performance» according to the New Oxford American Dictionary.

- a description is produced to focus in the relation between the performance, the lyrics and the sound is a semantic whole and give us a picture of what the composer might have envisioned at his time; e.g. the description of the costumes needed for the performance related to the partition. This is the **symbolic granularity** that focus in the social and not-explicit connotations of the elements in the description.

Third, in Soa2m we recover the incompleteness of the description.

Any director or crew in the performing arts is forced to describe all things in every detail. It depends on the initial investment in the project, the size of the team and the wanted expression.

Sometimes every aspect is covered, sometimes not. Sometimes only a punctual checklist is enough for launching a project, sometimes in a big production everything must be calculated and described.

In result, to fill the data model proposed in the Soa2m situation is evolutive and optional. While the general structure of the data is followed and respected, null values may be accepted.

In consequence, the proposal is a methodology for the annotation of generative processes and entities used to create situations of multimodal interaction and not a set of rules that must be followed if we expect some result. In other words, this is a design pattern proposal to submit to a discussion in the open standards and research communities.

Proposal: The Soa2m situation is an ontology design pattern

The Soa2m situation is a recursive method of description. This means that in the model, almost everything can be a situation, as we already showed in figures [**Figure 2.2.16**] [**Figure 2.2.17**] [**Figure 2.2.18**] and [**Figure 2.2.19**].

To begin, every Soa2m situation has a facet. [**Figure 2.2.22**] Facet analysis synthesizes complex descriptions from atomic elements; this is a set of fundamental categories (appropriate to the domain of multimodality) and their combination according to synthesis rules. In Soa2m a facet corresponds to an agent interest, this represents *for who* we will describe the situation with its interest on mind.

Proposal: A situation with a facet Layer

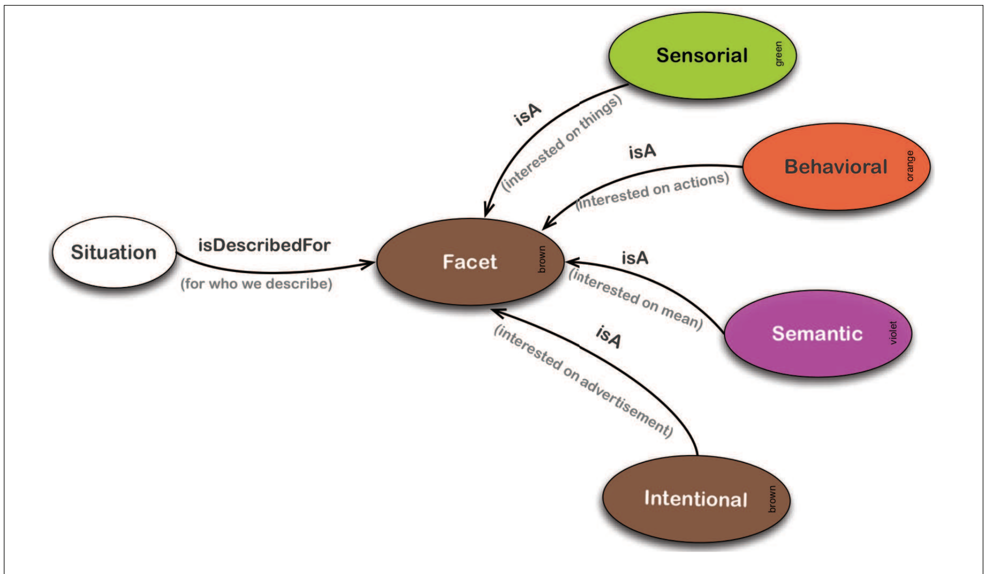


Figure 2.2.22: The Facet Layer in the Soa2m situation

In Soa2m facets can be inclusive. If we describe the situation according to the **sensorial facet** nothing else will be needed, but if we describe the situation according to the **behavioral facet**, chances are that we can use a **sensorial facet** also described somewhere. [**Figure 2.2.22**]

For example, the description of the acoustic *mode* in the **sensorial facet** (line 400) will list the participants in the situation at a low level: the phenomenon.

The description lists the participants -receptor (line 420), emitter (line 421), perceivedSound (line 425), attentive (line 419)- and its relations -listening (line 422), duringCapture (line 424)-:

```
395  <owl:NamedIndividual rdf:about="&mode;acoustic">
396     <rdf:type rdf:resource="&mode;mode"/>
397     <rdf:type>
398        <owl:Restriction>
399           <owl:onProperty rdf:resource="&Soa2mSituation;hasFacet"/>
400           <owl:allValuesFrom rdf:resource="&Soa2mSituation;sensorialFacet"/>
401        </owl:Restriction>
402     </rdf:type>
403     <rdf:type>
404        <owl:Restriction>
405           <owl:onProperty rdf:resource="&Soa2mSituation;hasGranularity"/>
406           <owl:allValuesFrom rdf:resource="&Soa2mSituation;relationalGranularity"/>
407        </owl:Restriction>
408     </rdf:type>
409     <rdfs:comment xml:lang="en">
410        [ INTENTION: To classify things based on the frequency range of hearing.| DEFINITION:  Quality
411        used to classify the things perceived according to the mechano-sensitivity of molecules in the world.
412        This is the generated (with the vocal cords or the body) or perceived (through the ears) frequency
413        of oscillation of waves ]
414     </rdfs:comment>
415     <Soa2mSituation:hasDescription rdf:resource="&mode;acousticDescription"/>
416  </owl:NamedIndividual>

417  <owl:NamedIndividual rdf:about="&mode;acousticDescription">
418     <rdf:type rdf:resource="&mode;modeDescription"/>
419     <Soa2mSituation:hasHowExecution  rdf:resource="&mode;attentive"/>
420     <Soa2mSituation:hasWhoActor  rdf:resource="&mode;receptor"/>
421     <Soa2mSituation:hasWhoOriginator  rdf:resource="&mode;emitter"/>
422     <Soa2mSituation:hasHowPerformance  rdf:resource="&mode;listening"/>
423     <Soa2mSituation:hasWhere  rdf:resource="&mode;distant"/>
424     <Soa2mSituation:hasWhen  rdf:resource="&mode;duringCapture"/>
425     <Soa2mSituation:hasWhat  rdf:resource="&mode;audible"/>
426  </owl:NamedIndividual>
```

Nevertheless, in the Soa2m situation this is not mandatory, and the decision will depend on the application needs and the required expressivity of the description.

Given that the situation is a data model based on semantic web ontologies and reasoning technologies, the amount of inferences do not depend on the completeness of the data: knowledge can be produced with incomplete descriptions.

Proposal: A situation with a granularity Layer

In addition to the facet, in Soa2m every situation is described according to the focus that we put in the situation: the granularity. [**Figure 2.2.23**].

The description can be fine-grained and then we focus in a **punctual granularity,** coarse-grained and then we focus in a **relational granularity** and finally the description can be unbounded-grained and the we focus in the **symbolic granularity**. [**Figure 2.2.23**]

The **punctual granularity** focuses on enumeration, while the **relational granularity** focuses on the connections.

For example, in the description above (line 406) the description is made for some agent interested on the sensorial things -the **sensorial facet**- but it is focused on the connections between these things.

These are the emitter and receptor roles, the listening activity that implies at least two things connected or the distant space that implies a referential scale (two related distant things). This corresponds to the **relational granularity** point of view.

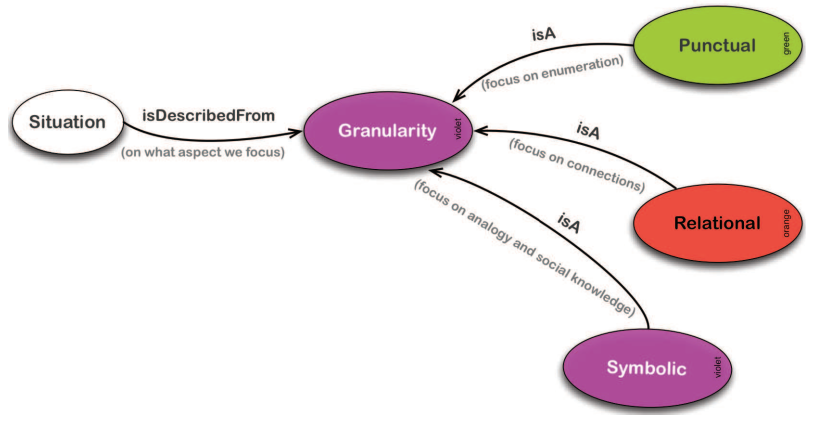


Figure 2.2.23: The Granularity layer in the Soa2m situation

Thus, if the same description is made in a **punctual granularity** it must enumerate sensorial things:

```
395  <owl:NamedIndividual rdf:about="&mode;acoustic">
396    <rdf:type rdf:resource="&mode;mode"/>
397    <rdf:type>
398      <owl:Restriction>
399        <owl:onProperty rdf:resource="&Soa2mSituation;hasFacet"/>
400        <owl:allValuesFrom rdf:resource="&Soa2mSituation;sensorialFacet"/>
401      </owl:Restriction>
402    </rdf:type>
403    <rdf:type>
404      <owl:Restriction>
405        <owl:onProperty rdf:resource="&Soa2mSituation;hasGranularity"/>
406        <owl:allValuesFrom rdf:resource="&Soa2mSituation;punctualGranularity"/>
407      </owl:Restriction>
408    </rdf:type>
409    <rdfs:comment xml:lang="en">
410      [ INTENTION: To classify things based on the frequency range of hearing.l DEFINITION:  Quality
411      used to classify the things perceived according to the mechano-sensitivity of molecules in the world.
412      This is the generated (with the vocal cords or the body) or perceived (through the ears) frequency
413      of oscillation of waves ]
414    </rdfs:comment>
415    <Soa2mSituation:hasDescription rdf:resource="&mode;acousticDescription"/>
416  </owl:NamedIndividual>

417  <owl:NamedIndividual rdf:about="&mode;acousticDescription">
418    <rdf:type rdf:resource="&mode;modeDescription"/>
419    <Soa2mSituation:hasHowExecution  rdf:resource="&mode;acousticSensing"/>
420    <Soa2mSituation:hasWhoActor  rdf:resource="&mode;ear"/>
421    <Soa2mSituation:hasWhoOriginator  rdf:resource="&mode;vocalCords"/>
422    <Soa2mSituation:hasHowPerformance  rdf:resource="&mode;perception"/>
423    <Soa2mSituation:hasWhere  rdf:resource="&mode;concert"/>
424    <Soa2mSituation:hasWhen  rdf:resource="&mode;noon"/>
425    <Soa2mSituation:hasWhat  rdf:resource="&mode;waveOscillation"/>
426  </owl:NamedIndividual>
```

In contrast, the focus on the **symbolic granularity**, will orient the description to the use of analogic or social (domain) knowledge.

Then, in the example above in line 420 we can replace «ear» with «inputMC», in line 422 we can replace «vocalCords» with «user», in line 425 «waveOscillation» with «message», in line 424 «hall» can be replaced with «outdoor»... and so on.

As we can see in the example, one granularity can be sufficient to describe a situation, depending on the facet and the description needs. But the superposition of the three granularities result in a more expressive description for each element of the description.

Proposal: A situation with a description Layer

To finish, we propose a layer covering the participants description. The description is a structure covering three types of information: the Who, the How and the Which. [**Figure 2.2.24**] It corresponds to the device, user and domain models studied in section 1.2.2 Participants in a Multimodal System ant the usage, time and space-time situations studied in section 1.2.3 The Interaction Context of a Multimodal System.

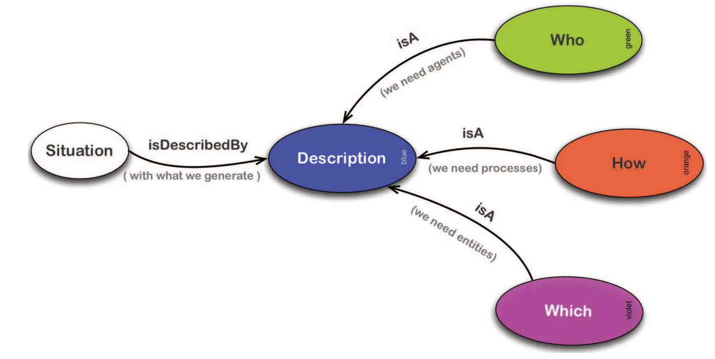


Figure 2.2.24: The Description layer in the Soa2m situation

The ***Who*** layer, describes the agents participating in the description. Depending on the granularity of the description, it can be populated with **actors** (direct agentive participants), **originators** (direct agentive participants influencing actors) and **inspirators** (indirect agentive participants). [**Figure 2.2.32**] For example, <USER, SPEAKER, FORM> is the description of ***Who*** participates in a situation of filling of a form by voice. We need an active user that will fill the data, a speaker who reproduces the instructions and a form structure representing the steps to follow: the form is an indirect agent that gives a reason to the action of the other two agents. In a real world description, for example, the **actor** can be the victim, the **originator** can be the robber and the **inspirator** the intellectual author of the crime.

The ***How*** layer, describes the processes in the situation. Depending on the granularity, it can be populated with **executions** (the description of a task), **performance** (the «instantiation» of the practice) or the **intention** (the expression of a plan with a desired outcome). [**Figure 2.2.32**] For example <PLAY, CRESCENDO, ALERT> is the description of *How* a wake-up situation can be generated. We will play something (sound, image, vibration), performed in crescendo[162] putting an accent in the goal of alerting the user by looping during 10 minutes.

Finally the ***Which*** layer, describes the entities needed for the situation: **where** (a place), **when** (a temporal region) and **what** (an entity). [**Figure 2.2.32**]

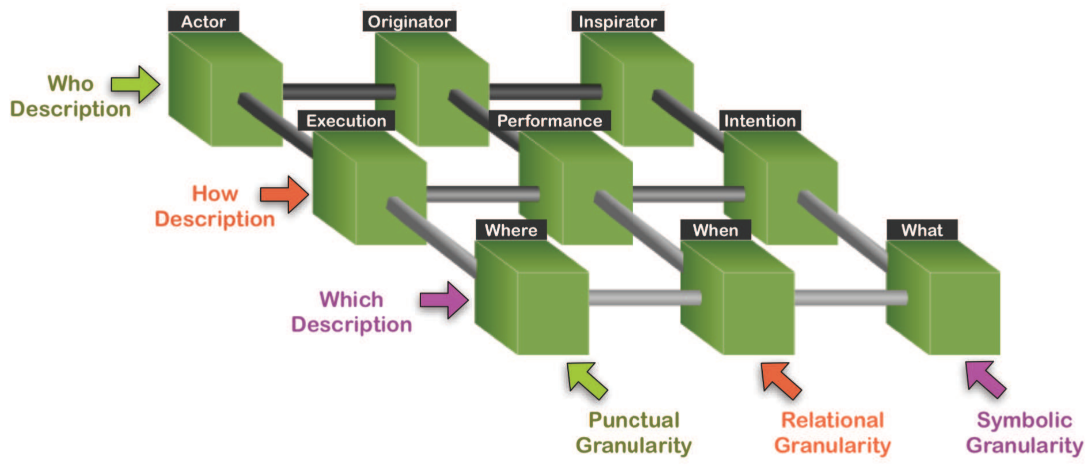


Figure 2.2.25: Description and Granularity for a Sensorial Facet in the Soa2m situation

[162] A performance during which some properties of the performance gradually increases: for example the rhythm or the volume or the size.

The **where** describes the space-time situation presented in section 1.2.3.3, following, for example, the Ontology for Geographical Information Systems of [Frank 2003]. The **when** describes the temporal situation presented in section 1.2.3.2., following, for example the valid, transactional and symbolic time model and the temporal operators of [Allen et al. 1985], [Freksa 1992] [Ultsch 1996] and [Möerchen et al. 2010]. And the **what** describes entities (any particular thing) of the situation.

In result, the Soa2m situation description can be a relational table in one dimension (the **sensorial facet**) as in the example in [**Figure 2.2.25**] or a hypercube of metadata in 2 or 3 dimensions or 4 dimensions [**Figure 2.2.26**] depending on the number of facets that the Modality Component service provider choose to describe e.g, the attribute **what** can be described in one or various facets [**Figure 2.2.26**].

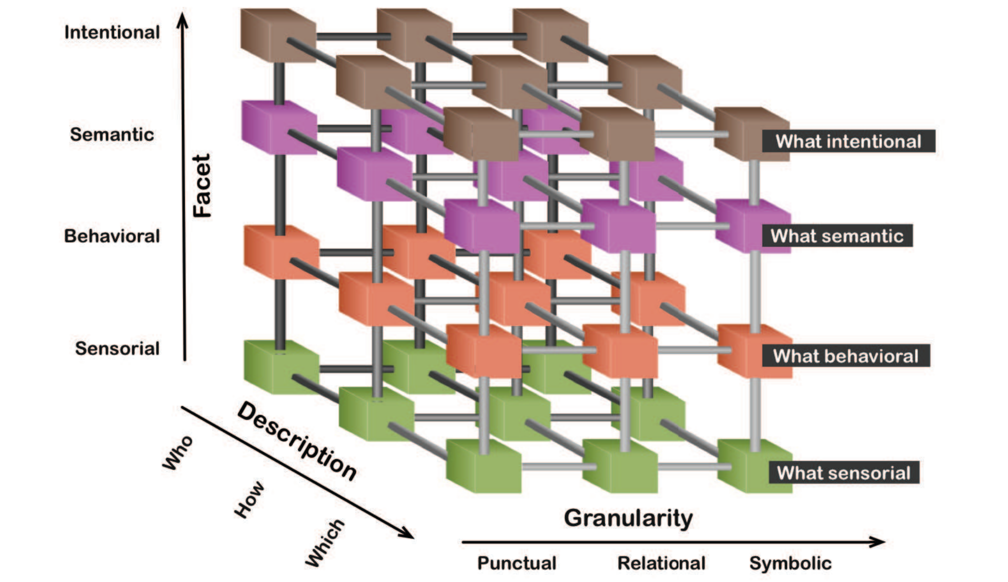


Figure 2.2.26: The Soa2m situation hypercube

Conclusion

To sum up, in this section we presented a situation data model to describe Modality Components services for the discovery and registering processes.

The Soa2m situation, is proposed as a basic support for the annotation mechanism explained is section 2.2.2.3 Description of MC Services with WSDL documents with usage ontologies.

The situation data model is structured on three axis: an axis concerning the approach of the situation (the facet), a second axis concerning the focus of the situation (the granularity) and a third axis concerning the annotation of the situation (the description).

The data model is a result of the analysis of two models . One comes from the multimodal research, the What Which How Then (WWHT) model [Rousseau 2006] describing the information allocation and instantiation in multimodal intelligent presentations. The second, the Event-Model-F [Scherp et al. 2009] comes from the multimedia and semantic web research, and was designed to annotate media documenting real-world events. Finally the data model is the result of the transposition of pre-production methods to describe situations in the performing arts.

The intent with the Soa2M situation is to contribute to the description of the usage situation for multimodal Modality Component services, by providing a starting point for an agreement in a generic and standardized data structure to enrich the MMI Framework & Architecture recommendation around the Data Component and its interfaces with a decision engine. The situation data model is founded in the notion of situation coming from DOLCE-Lite and is the basis for the implementation of an ontology as a design pattern for context description in multimodal systems.

2.2.4 Conclusion

This section presented the semantic aspects proposed to enrich the Soa2m Architecture with metadata annotations coming from the semantic web services community with the objective to enhance the discovery, registering and management of services of *multimodal systems* created according the Soa2m architecture proposal.

First, we presented the requirements for the management of modality component services to draw the limit for our semantic proposal. This requirements were distributed on several points: the advertisement of services, their discovery and their registration.

-The MC Services **advertisement** is the process to declare an announcement or to request an announcement of services. The advertisement can be initiated by the Controller Component or declared by the Modality Component to the Advertise Manager using a Soa2m service provided for this task.

Four cases of advertisement were presented: in the first case, the announcement is provided at design-time by a client application to be used and updated at run-time; in the second case, each Modality Component service wishing to join the multimodal system, announces its availability at design time, then the Controller Component confirms their availability and presence at load-time or run-time; in the third case all the descriptions for all the Modality Components in the system are provided at design time once for all and finally, at run-time, the Controller Component can «scan» the network to find available services.

On the other hand, the announcement presents the service and is based on three types of descriptions enriched with semantic web technologies: a data structure for tags, a manifest describing properties and relations and a web services description linked with metadata to describe the processes, restrictions and situation of usage. These descriptions increase in difficulty and working load to fulfill different levels of needs. Thus, the data described by these three description formats can be structured in four categories: identity, behavioral, semantic and intentional data.

Thus, the annotation proposal in Soa2m must be generic to the point to become a real contribution in services announcement as described in this requirement.

-The MC Services **discovery** is the automatic detection of non-advertised MC Services at load and run-time. Four types of discovery are identified as requirements in Soa2m: the *intention* discovery (based on the motivation for the service), the *semantic* discovery (based on the commitments of the service), the *behavior* discovery (based on the processes provided by the service) and the *capacities* discovery (based on the attributes of the service). In context-aware or intelligent multimodal systems, the first two can be required, while in most of the cases the last two types of discovery are enough. In Soa2m, *intention* and *semantic* discovery are supported by rich descriptions based on semantic web ontologies and *behavior* and *capacities* discovery are supported by tagging mechanisms and manifests.

Services discovery can be executed in multiple ways: fixed discovery is based on hard coded information at design-time and is used to confirm this information at load-time; mediated discovery is based on the use of directories provided by networking technologies; active discovery implies multi-cast, broad-cast or any-cast initiatives from the Controller Component to discover Modality Component services and finally, in passive discovery the Controller Component will listen advertisement announcements coming from the services.

In result, the semantic annotation of Modality Component services must be extensive enough to become a real asset in this four types of discovery and their most common implementations.

-The MC Services **registration** is the process to store information about the services according to a precise data model in a registry. In Soa2m a registry is a service provided by the Advertise Manager, storing and disseminating the information about the *multimodal system*.

Two methods of registration can be implemented: a push mechanism or an adaptive pull mechanism. The validity of each registered information can be limited or unlimited. If the validity is limited, the procedure is called «soft-state» registering, and a mechanism for renewal must be implemented. In Soa2m, to address soft-state registering a timeout data structure is proposed, carrying the information about the registering state of the service. The same structure is also used to declare the unlimited registration, called «hard-state» registering.

In order to renew the registration, the Soa2m architecture extends the MMI Framework & Architecture recommendation by adding two events to the Events Life-Cycle protocol: the checkUpdate Event and the UIUpdate Notification. This two events, used in conjunction with periodical requests ensure the automatic updates of the state of Modality Component services, and their registration.

On the other hand, the registry is constructed according a data model structuring the information according four facets: the **sensorial**, the **behavioral**, the **semantic** and **intentional** facets.

Every item in the registry is identified according to an intentional scope naming convention proposed to enhance discovery at bootstrapping by providing high level semantic information about the service in the discovery announcement through its name.

Finally, the semantic orientation of the registry can collaborate on its distribution. A registry can be centralized, where a rich semantic data model can enhance services search and state handling; but it can also be distributed and in this case semantics can help to support the distribution, for example, to consolidate federated registries according with multimodal data or composition design spaces like the CARE properties. In result, the semantic annotation of Modality Component services must be expressive enough to become an improvement for the structuring, searching and registering needs in multimodal systems.

The presentation of these three issues (advertisement, discovery and registering) allow us to specify nine requirements for the functional responsibilities for the Advertise Manager and the Data Component of the Soa2m architecture that are currently in the process of discussion and specification in the W3C Multimodal Working Group. These responsibilities are:

- To register semantically annotated services.
- To support ontology matching (for context-aware implementations).
- To support periodical updates of registers and states.
- To handle timeout structures in soft-registering cases.
- To perform a gross selection of services based on high level information on the services name.
- To manage a generic welcome group for unknown categories of services' scopes.
- To disseminate information about services through a directory.
- To store the registry according a semantic data model implemented in the Data Component.
- To provide the interfaces for the indexing and querying mechanisms.

The first and second responsibilities are linked with the semantic annotation support. Therefore the second part of this section explained the contribution of the Soa2m project concerning this topic.

To begin, we presented the annotation procedures proposed in Soa2m, covering the intelligent use of some existent technologies to fulfill our purpose to annotate not only functional behavior but also contextual and usage conditions in a synergic[163] way. In result we propose three formats and methods of annotation (data structures, semantic manifests, extended web service description documents) adapted to different needs. These procedures respond to the three requirements explained above (advertisement, discovery, registering) and are compatible with the Soa2m architectural perspective.

Then, we presented the data model to be used in association with these annotation methods. The proposal is based on the premise that services description covers two approaches: the service functioning and the service usage. While the first approach concern IOPE properties[164], the second is guided by the notion of situation. In result, we proposed a multimodal annotation glossary for advertisement and discovery and an ontology knowledge model for discovery and registering.

-In Soa2m three **annotation methods** are proposed: YAML data structures, RDF semantic manifests, and WSDL documents extended with ontologies.

The first method is based on a data serialization language designed to be human-friendly the «YAML Ain't Markup Language» (YAML) [Ben-Kiki et al. 2009]. By the use of collections and scalars it provides extremely readable data structures very helpful for a loosely controlled tagging process, for example, for RESTful services. Thus, combined with a precise controlled vocabulary or a glossary, the annotations with YAML can describe with low effort input and output interfaces, identification and behavior data and even some non-functional data like the situation of use. In this section we provided examples of annotation of these properties and an introduction to the data model defined in the Soa2m glossary.

The goal of this format is the description of services with tags in well known and controlled multimodal systems, for example, in multimodal systems with low needs in extensibility and dynamics.

The second method, the semantic description with RDF manifests is based on resource description languages coming from the semantic web standards: RDF [W3C-RDF 2004] and RDF Schema [W3C-RDFS 2004]. By the use of these markup languages, we propose an annotation method that allows the description of classes, properties and relations in the information advertised by the service. In the simpler cases, this description provides a rich description of dataTypes, in more complex cases, e.g. for context-aware needs, it can be aligned with an inheritance structure like a Faceted Thesaurus[165], enumerating a taxonomy of Modality Component services , their relationships and restrictions.

The third method, extended WSDL [W3C-WSDL 2007] annotation, is based on the use of the *modelReference* attribute linking the document to a data model [W3C-SAWSDL 2007]. We propose the synergic use of this attribute to describe functional and non-functional data, depending on the element that is linked to the data model and conventionally tagged with two prefixes: «ser» and «use».

In this way, WSDL elements that already describe functional information, like the service name, binding, interface, operations and data types, can be extended with complementary semantic data. For example, while the WSDL can enumerate the inputs of an operation and its data, a complementary «ser» ontology can complete this information describing the effects of an input in the final result. This is an information of a higher lever that can not be described with the tools provided by the WSDL standard.

163 Synergy is two or more entities functioning together to produce a result that is not obtainable independently. It means "working together". For example, in nomadic applications (always affected by the changing context) to avoid disruptive interaction is an important issue. In these applications, user interaction is difficult, distracted and less precise. A synergic description is a description that informs about the functional behavior and about the influencing context that limits this functioning. In this way, a well-founded discovery mechanism can enhance the fusion process for target groups of users experiencing permanent or temporary learning difficulties or with sensorial, emotional or social impairments.

164 Inputs, Outputs, Preconditions and Effects are the IOPEs properties

165 A Thesaurus is a set of terms organized according a limited number of relationships. A faceted classification structure entities by picking one term from a facet to describe the document along all the different axes. This would then describe a document from many different perspectives.

Other example, is the information about the service's operations, described in WSDL by a method of request, a name or a data type can be completed with the semantic meaning of the provided task, in terms of the situation of use. In this case, a complementary «use» ontology can describe the conditions of use of the service, for example, only in a private space in working hours.

The design and implementation of these data models in the form of ontologies is ensured by two semantic web tools: the OWL-S markup language [W3C-OWLS 2004] can be used to produce the service ontologies identified with the prefix «ser» and the DOLCE-Lite foundational ontology [Gangemi et al. 2002] can be used to produce the usage ontologies identified with the prefix «use».

Thus, the service ontologies are proposed to fulfill the requirements on behavior and capacities discovery and registering while the usage ontologies are proposed to collaborate on capacities discovery and registering and to fulfill the requirements on intention and semantic discovery and registering.

The goal of this format is the description of services for context-awareness needs. The working load and complexity inherent to this annotation method is counterbalanced by the expressiveness of the descriptions and their collaboration on tasks adapted to the user. These descriptions are combined with a coherent data model describing the usage situations.

With this objective on mind, we proposed in Soa2m a knowledge data model to support the implementation and use of situation ontologies.

-In Soa2m two approaches for **data modeling** are proposed: a multimodal annotation glossary for advertisement and discovery; and a knowledge model for discovery and registering.

The Soa2m Glossary is divided in two parts. The first set of properties is designed to enhance the identification of the service in the bootstrapping announcement or in the discovery process. It concerns the positioning of the service in an inheritance or subsumption (be part of) structure. Then, this part of the Glossary defines de foundational notions in multimodal systems (*mode, modality, medium)* and defines the name and affiliation properties of Modality Component services. These four terms, name, affiliation, mode, modality; are the minimal properties that can be annotated with data structures or manifests in Soa2m.

The second set of properties is designed to support the discovery and registration process on the selection of services according to their behavior and life-cycle. It concerns the Soa2m architectural design pattern used in the service implementation which provides semantic information about its behavior , its state, its validity, its advertised skills and its behavior concerning the user privacy. These five terms, Life, State, Type, Skill, Privacy, are the second part of properties that can be filled in data structures or manifests to describe the service behavior and states.

These two set of properties (non-exhaustive) come from the analysis of six taxonomies: [Foley et al. 1984] [Buxton 1986] [Mackinlay et al. 1990] [Arens 1993] [Bernsen 1993] [Bellik 1995] and the inspiring proposals of two multimodal data proposals: the properties lists of input devices [Bouchet 2006] and the Devices and Task component interfaces [Serrano 2010].

Finally this Soa2m basic (and extensible) glossary for discovery and registering is proposed as a starting point for the discussion around the MMI Framework & Architecture standardization of Modality Components.

On the other hand, the Soa2m situation knowledge model for discovery and registering is designed to address the current and future needs of intelligent multimodal systems implemented with the MMI Framework & Architecture recommendation.

To begin, it addresses the discussion about the required interfaces and behavior in the Data Component, its relations with the Controller Component and its collaboration in the semantic *fusion* and *fission* processes.

But it also addresses the discussion about the context handling required in multimodal systems, its limits and its contribution to the intelligent management of the global state of the system and the application.

The model is inspired on the **What Which How Then** (WWHT) model [Rousseau 2006] for the intelligent instantiation of multimodal presentations and the **Script-Breakdown** procedure for describing situations in the performing arts. And it inherits of the annotation approach proposed by the **Event-Model-F** [Scherp et al. 2009] to document events for real world understanding, semantic searching of media and reality mining[166].

The Soa2m situation is a facetted ontology, that classifies the situation into four layers of knowledge -the *facets*- depending on the interests of the agent recipient of the description: the **sensorial facet** lists things needed by the recipient, the **behavioral facet** describes behaviors of agents and things, the **semantic facet** describes the social (domain) meaning of agents or things and the **intentional facet** that describes the advertised intention of agents or things.

With these facets, the situation is explained according to the perspective and interests of the final agent that will use it. For example, if the description is intended for a state management service, intentional data can not be necessary. If the description is intended for a interaction manager, behavioral or sensorial data can be pertinent while semantic data will be more adequate for a decision engine.

On the other hand, the Soa2m situation is also described with a given *granularity*: the **punctual granularity** focuses on enumeration, the **relational granularity focus**es on the connections and the **symbolic granularity**, is oriented to the use of analogic knowledge.

With these granularities, the situation is explained according to a precise point of view. For example, if the final agent processes detailed information like dataTypes, names, coordinates, low level phenomena, a symbolic granularity will be meaningless. But if the agent needs gross information to infer a global situation before handling the specific details, the symbolic description will be more pertinent.

Finally, the Soa2m situation is annotated by a description of three contextual conditions: the **Who**, the **How** and the **Which**. The result of this proposal is a structure of annotation properties [Table 2.2.7] and an OWL ontology based on DOLCE-Lite.

		GRANULARITY		
		Punctual	Relational	Symbolic
DESCRIPTION	Who	ACTOR	ORIGINATOR	INSPIRATOR
	How	EXECUTION	PERFORMANCE	INTENTION
	Which	WHERE	WHEN	WHAT

Table 2.2.7: Description properties according to granularities in Soa2m.

Evaluation Criteria for the Soa2m data model.

To evaluate the Soa2m situation data model, some of the criteria presented in section 2.1. The Soa2m Architecture can be used. A data model can be evaluated against: its coverage of a particular domain (**RD7** Completeness) and the richness (**RD2** Flexibility), complexity (**RD4** Reusability) (**RD5** Modifiability) and granularity (**RI3** Configurability) of that coverage; the specific requirements and data sources it was developed to address; and formal properties such as the consistency of the data (**RD8** Consistency) and the representation language in which it is modeled.

Evaluation Criteria for the Soa2m ontologies

In the case of modeling with ontologies, the evaluation includes [Guarino 2004] aspects of ontology validation and verification: structural (**RI4** Buildability), functional (**RI5** Automation) (**RI6** Integrability) (**RD1** Extensibility) (**RR2** Performance) (**RR5** Scalability), and usability issues (**RR1** Interoperability) (**RI1** Accessibility) (**RI2** Usability).

166 The collection of environmental data pertaining to human social behavior.

Design-Time Non-Functional Requirements	**RD1.** Extensibility	Proposition of a faceted model extensible with other agent interests.
	RD2. Flexibility	As a data model linked to the semantic web, it inherits of the extensibility ensured by the web of data approach.
	RD4. Reusability	Its generic approach allows the annotation of multiple real-world situations . It is not domain-dependent.
	RD5. Modifiability	The data model is generic enough to support the modification of properties with another domain-dependent semantics.
	RD7. Completeness	Proposition of a hypercube structure that can be extended or break-down into new facets, granularities or elements of descriptions. Nevertheless the current proposal is sufficient to cover a rich quantity of use cases.
	RD8. Consistency	The pursuing of the standardized effort of description (OWL-S, DnS Plugin) based on foundational ontologies for services and context handling. The proposal is consistent also by using the same conceptual model of the Soa2m architecture.
Run-Time Non-Functional Requirements	**RR1.** Interoperability	Proposition of use of web semantic standards and standardized or well known notions e.g. the IOPE properties, the CARE properties, the Allen operators the proxemics distances.
	RR2. Performance	***NOT ADDRESSED in the Data Model Proposal***
	RD8. Scalability	As a design pattern implemented in a ontology the data model can be reduced or increased depending on the needs. e.g. one only facet with a single granularity.
Interaction-Time Non-Functional Requirements	**RI1.** Accessibility	Proposition of a data model translating complex concepts used in ontology design to common use notions. Extensive description of each of these concepts with illustrative images coming from real-world experience. Three levels of complexity in the data description method and two levels of complexity in the data model. Graphic explanation of the complex notions of the model. It also represents different point of views of situations and mental models of the world (positivist, hermeneutic, phenomenological).
	RI2. Usability	Naming conventions enhancing learnability and memorability.. Three levels of difficulty (tags, manifests, ontologies) for different users.
	RI3. Configurability	The data model is designed to be extended, reduced or incomplete if needed.
	RI4. Buildability	The data model is a good support to the creation of an authoring tool for annotation of services. STructures can be build easily according the different relational tables offered.
	RI5. Automation	The data model can support automation in the annotation of services, offering data structures easy to parse and well supported by multiple tools (XML oriented).
	RI6. Integrability	The data model is designed based on two well known ontologies. This allows the Soa2m situation description to be integrated and aligned with existent projects.

Table 2.2.8: The Soa2m situation data model evaluation.

[**Table 2.2.8**] presents the evaluation of the Soa2m situation data model according to these requirements. As expected, the main conclusion that we can reach is that as the model is flexible, extensible, scalable, interoperable, consistent etc, it is difficult to evaluate its performance, accessibility and usability in the current conditions of the semantic web[167]

In this section we presented the semantic annotation formats and the data models that was a basis for the implementation of the tools for the Soa2m architecture.

The next section will list these results under the light of the contribution for the discovery and registering of Modality Component Services according with the MMI Framework & Architecture recommendation.

167 Technological advances are needed in the available tools for annotation to enhance performance and real-time support. The current tool also demand very specific competences: semantic annotation must be more accessible. This is a difficult task to be done by specialists to share with a large community.

2.3 The Soa2m Libraries implementation.

In this section we will use scenarios to motivate three types of user requirements and to present the application of the situation model proposed, the semantic services approach and the extensions of the architecture to three scenarios in order to introduce the libraries implemented during this thesis.

The first scenario is adapted from a common scenario in pervasive systems. This scenario describes how a conference assistant can use context data to propose adapted services to the user in controlled spaces. The second scenario is based on ethnographic studies of situations executed during three years of *in-situ* reappropriation[168] of spaces. This scenario describes the overlapping of situations and the role of discovery and registering for multimodal services instantiated in polyvalent spaces with a precise timing and coordination. The third scenario is based in a therapeutical project of the Elikya Center for the integration and recovery of child soldiers[169] and is more complex. The healthcare staff in this setting work with disabled patients not only at the physical level but also and mostly at the social level. In this scenario physical adaptation and the management of hypersensitive situations is crucial because it could affect the integration and recovery of the patients.

Outline

In the following section we will not focus on the scenario description but we will list the implementations produced in Soa2m as a result of the analysis of these situations under the light of services discovery and registering.

2.3.1 Scenario 1- The controlled situation: The conference.

Imagine a conference room where a series of seminars will take place. People enter and leave the conference room before, after and during the lectures. Before the meeting, the room access manager, sensing no human presence, is running in its sleep mode. When the system activates it instantiates the default services for the scheduled situation in the room: the outline service, the notification service, the guiding service. The chair of the meeting selects the agenda instance of the outline service, and submits the agenda content with the presentation slides. The outline service composes an animated conference program and synchronizes its evolution state with the slides information. The outline service publishes this conference program (by default in the HDTV display and also available by internet or in nearest display devices), the conference notes service (for phone or pc text, image or audio input) and the washroom location guide service as new services in the conference room.

2.3.1.1 The Web Services generation and annotation library

Quick introduction of semantic web services in Soa2m

Web services are a means to provide an interoperable platform for application integration using web technologies. The capability to find useful services depends on the matching of their descriptions possibly enhanced with some ontological knowledge. The term «Semantic web services» stands for the automation of service's life-cycle tasks such as discovery, selection, composition, and instantiation of suitable services. This task is accomplished by making services themselves machine processable.

The description of web services in a machine-understandable fashion is expected to have a great impact in the areas of e-commerce and enterprise application integration, as it can enable dynamic and scalable cooperation between different systems and organizations.

WSDL 2.0 is and semantic annotation not supported by web services tools

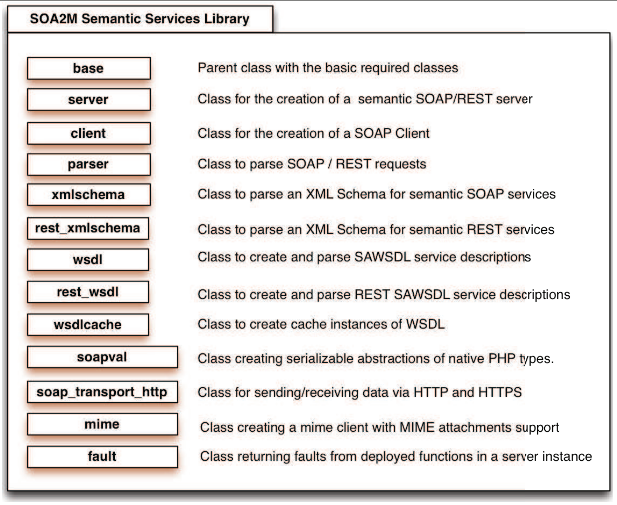
Nevertheless, the standard most used for the description of web services, WSDL 1.1 has one significant drawback: the semantic annotation with ontologies by the use of a dedicated attribute (*referenceModel*) is not supported[3], and the current recommendations WSDL 2.0 / SAWSDL, are not compatible with the existent tools for the service description and annotation.

168 An existing space may outlive its original purpose and the «raison d'etre» which determines its forms, functions, and structures; it may thus in a sense become vacant, and susceptible of being diverted, reappropriated and put to a use quite different from its initial one.
169 http://invisiblechildren.com/program/rehabilitation-project/

Result: A library for the management of semantic services

For this reason, the first step in the Soa2m project at the implementation level was the development of a semantic services description library supporting the semantic annotation proposed by SAWSDL.

The Soa2m Semantic Services Library [Figure 2.3.1] is based in NuSOAP, a rewrite of SOAPx4, provided by NuSphere and Dietrich Ayala. It is a set of PHP classes that allow developers to create and consume web services based on SOAP 1.1, WSDL 1.1 and HTTP 1.0/1.1.

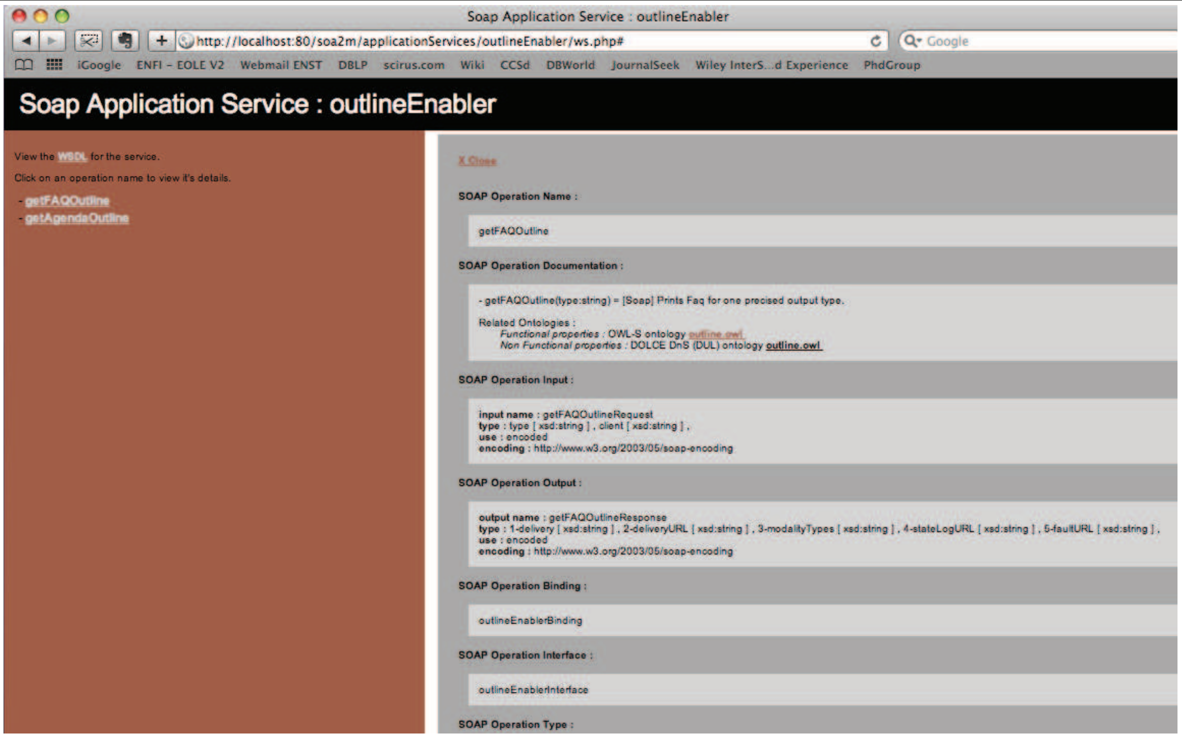


SOA2M Semantic Services Library

base	Parent class with the basic required classes
server	Class for the creation of a semantic SOAP/REST server
client	Class for the creation of a SOAP Client
parser	Class to parse SOAP / REST requests
xmlschema	Class to parse an XML Schema for semantic SOAP services
rest_xmlschema	Class to parse an XML Schema for semantic REST services
wsdl	Class to create and parse SAWSDL service descriptions
rest_wsdl	Class to create and parse REST SAWSDL service descriptions
wsdlcache	Class to create cache instances of WSDL
soapval	Class creating serializable abstractions of native PHP types.
soap_transport_http	Class for sending/receiving data via HTTP and HTTPS
mime	Class creating a mime client with MIME attachments support
fault	Class returning faults from deployed functions in a server instance

Figure 2.3.1: The Soa2m Semantic Services Library

The Soa2m Semantic Services Library updates this library to SOAP 1.2 and WSDL 2.0 and adds the support of SAWSDL, the annotation of RESTful services and the management of JSON Requests. The Soa2m Semantic Services Library allows the creation of a semantic services server handling Soap and Rest requests (Json) [**Figure** 2.3.3]. The soap client and server are then extended with parsing and serialization functionalities conform to the WSDL 2.0 standard and the SAWSDL standard. With this extension, the service, its bindings, its interfaces, its operations and its dataTypes can be annotated with semantic web ontologies.


Soap Application Service : outlineEnabler
http://localhost:80/soa2m/applicationServices/outlineEnabler/ws.php#
Google
iGoogle
ENFI – EOLE V2
Webmail ENST
DBLP
scirus.com
Wiki
CCSd
DBWorld
JournalSeek
Wiley InterS...d Experience
PhdGroup

Soap Application Service : outlineEnabler

View the WSDL for the service.

Click on an operation name to view it's details.

- getFAQOutline
- getAgendaOutline

X Close

SOAP Operation Name :

getFAQOutline

SOAP Operation Documentation :

- getFAQOutline(type:string) = [Soap] Prints Faq for one precised output type.

Related Ontologies :
Functional properties : OWL-S ontology outline.owl
Non Functional properties : DOLCE DnS (DUL) ontology outline.owl

SOAP Operation Input :

input name : getFAQOutlineRequest
type : type [xsd:string] , client [xsd:string] ,
use : encoded
encoding : http://www.w3.org/2003/05/soap-encoding

SOAP Operation Output :

output name : getFAQOutlineResponse
type : 1-delivery [xsd:string] , 2-deliveryURL [xsd:string] , 3-modalityTypes [xsd:string] , 4-stateLogURL [xsd:string] , 5-faultURL [xsd:string] ,
use : encoded
encoding : http://www.w3.org/2003/05/soap-encoding

SOAP Operation Binding :

outlineEnablerBinding

SOAP Operation Interface :

outlineEnablerInterface

SOAP Operation Type :

Figure 2.3.2: The Soa2m Semantic Services Documentation Interface

In addition to the soap WSDL serialization and deserialization according with a newer version of the standard, and the semantic extension for the WSDL elements; the Soa2m Semantic Services Library includes the REST serialization and deserialization of JSON requests and its XMLSchema. [**Figure 2.3.3**]. Finally, it generates a human-readable interface for the documentation of the services [**Figure 2.3.2**].

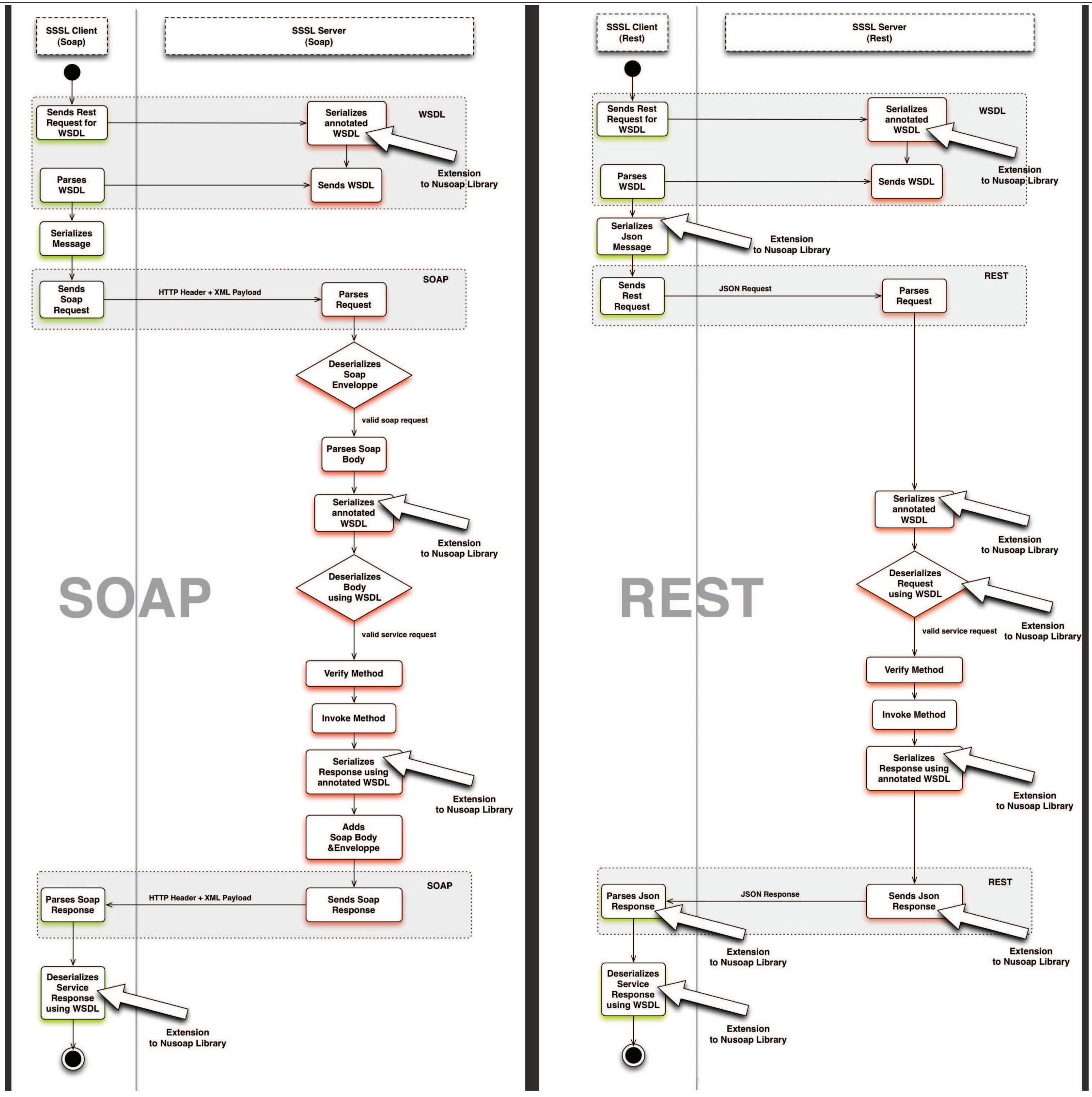

SSSL Client (Soap)
SSSL Server (Soap)
Sends Rest Request for WSDL
Serializes annotated WSDL
WSDL
Extension to Nusoap Library
Parses WSDL
Sends WSDL
Serializes Message
SOAP
Sends Soap Request
HTTP Header + XML Payload
Parses Request
Deserializes Soap Envelopppe
valid soap request
Parses Soap Body
Serializes annotated WSDL
Extension to Nusoap Library
SOAP
Deserializes Body using WSDL
valid service request
Verify Method
Invoke Method
Serializes Response using annotated WSDL
Extension to Nusoap Library
Adds Soap Body &Envelope
SOAP
Parses Soap Response
HTTP Header + XML Payload
Sends Soap Response
Deserializes Service Response using WSDL
Extension to Nusoap Library
SSSL Client (Rest)
SSSL Server (Rest)
Sends Rest Request for WSDL
Serializes annotated WSDL
WSDL
Extension to Nusoap Library
Parses WSDL
Sends WSDL
Serializes Json Message
Extension to Nusoap Library
REST
Sends Rest Request
JSON Request
Parses Request
Serializes annotated WSDL
Extension to Nusoap Library
REST
Deserializes Request using WSDL
valid service request
Extension to Nusoap Library
Verify Method
Invoke Method
Serializes Response using annotated WSDL
Extension to Nusoap Library
REST
Parses Json Response
JSON Response
Sends Json Response
Extension to Nusoap Library
Extension to Nusoap Library
Deserializes Service Response using WSDL
Extension to Nusoap Library

Figure 2.3.3: The Soa2m Semantic Services Library extension to Nusoap

The following is an example of use of the Library for a CocoatouchGreetingNotifier MC Service. The first step is to define the annotation requirements based on previously developed ontologies.

Annotation Requirements	**Service** **(CocoatouchGreetingNotifier)**	use:Greeting#Service		
	Operation **(getWelcomeGreeting)**	use:Greeting#Welcome		
	Input **(getWelcomeGreetingRequest)**	ser:Greeting#ServicePolicy use:Greeting#TemplateCode	ser:Greeting#ContentURL ser:Greeting#ClientKey	
	Output **(getWelcomeGreetingResponse)**	use:Greeting#MMWelcome ser:Greeting#Deliver	ser:Greeting#DeliverURL ser:Greeting#TypesList	ser:Greeting#StateLogURL ser:Greeting#FaultURL

Table 2.3.1: Annotation needs for the CocoatouchGreetingNotifier service

And the second step is to describe the service in a template structure. For example, the following code shows the annotation of the operation inputs and outputs with the template:

```
$doc =" - WelcomeGreeting (templateCode:string,contentURL:string,client:string) =
                                                            [Soap] Returns a Welcome greeting in a specific mode."
$operation_name = 'Welcome';
        $ws_class_dic[$operation_name] = array (
                $operation_name.SERVICE.'Request' => array(
                        'templateCode' => array( 'type' => 'str',
                                        'nillable' => 'true',
                                        SAWSDL_PARAMETER =>USE":".GREETING_ONT.'#TemplateCode',
                                        'restriction' => 'str',
                                        'length' => '8' ) ,
                        'contentURL' => array( 'type' => 'str',
                                        'nillable' => 'false',
                                        SAWSDL_PARAMETER =>SER":".GREETING_ONT.'#contentURL',
                                        'restriction' => 'str',
                                        'length' => '255' ) ,
                        'clientKey' => array( 'type' => 'str',
                                        'nillable' => 'false',
                                        SAWSDL_PARAMETER =>SER":".GREETING_ONT.'#clientKey',
                                        'restriction' => 'str',
                                        'length' => '10' ) ,
                SAWSDL_PARAMETER => SER.":".GREETING_ONT.'#ServicePolicy' ) ,
                $operation_name.SERVICE.'Response' => array(
                        'delivery' => array( 'type' => 'boolean',
                                        'nillable' => 'false',
                                        SAWSDL_PARAMETER =>SER.":".GREETING_ONT.'#deliver',
                                        'restriction' => 'str' ) ,
                        'deliveryURL' => array( 'type' => 'str',
                                        'nillable' => 'true',
                                        SAWSDL_PARAMETER =>SER.":".GREETING_ONT.'#defaultDeliver',
                                        'restriction' => 'str' ) ,
                        'modality' => array( 'type' => 'str',
                                        'nillable' => 'false',
                                        SAWSDL_PARAMETER =>SER.":".GREETING_ONT.'#typesList',
                                        'restriction' => 'str',
                                        'length' => '255' ) ,
                        'state' => array( 'type' => 'str',
                                        'nillable' => 'false',
                                        SAWSDL_PARAMETER =>SER.":".GREETING_ONT.'#StateLog',
                                        'restriction' => 'str',
                                        'length' => '10' ) ,
                        'faultURL' => array( 'type' => 'str',
                                        'nillable' => 'false',
                                        SAWSDL_PARAMETER =>SER.":".GREETING_ONT.'#Fault',
                                        'restriction' => 'str',
                                        'length' => '10' ) ,
                SAWSDL_PARAMETER =>USE.":".GREETING_ONT.'# MMWelcome' ) ,
```

The implementation is for the moment limited to a set of templates but as you can see in the example, a visual front end for annotation can be produced to simplify the task of filling these parameters for the description of the service.

Then, in a second step, the Soa2m Semantic Services Library will register the service in the server based on this description (line 19 to 29- code snippet in next page) , and generate the corresponding annotated WSDL.

The library avoids for the service provider the manual generation of the XML markup in WSDL, the restriction of datatypes with an xml schema and the creation of a SOAP server and a SOAP client capable to support this kind of semantic descriptions.

In conclusion, with the implementation of the Soa2m Semantic Services Library, we generate the first tool allowing the discovery and registering of semantically annotated services fully compliant with the current available standards.

```
/**
 * This is the register for an operation in a SOAP webservice
 * When successful, the following values are defined :
 * Method Name : the name of the PHP function, class.method or class..method
 * Input Interface : assoc array of input values: key = param name, value = param type
 * Output Interface : assoc array of output values: key = param name, value = param type
 * Operation Namespace : the element namespace for the method or false
 * Soap Action : soapaction for the method or false
 * Used Protocol : Optional (rpc|document) or false. When 'document' specified, parameter and return wrappers are created
 * Options : optional (encoded | literal) or false
 * Doc : optional Description to include in WSDL
 * EncodingStyle : optional (usually 'http://schemas.xmlsoap.org/soap/encoding/' for encoded) public
 * Exchange Pattern : in-only, In-out, out-only
 * Semantic Annotation:  SAWSDL_PARAMETER  or false
 *
 * @param server $server  SOAP Object
 */

function  getOperationRegister  ( $server,  $ws_class_dic, $operation_name, $doc )  {
$server->register($server->wsdl->currentOperation,
                        $ws_class_dic [ $operation_name ] [ $operation_name.SERVICE.'Request' ] ,
                        $ws_class_dic [ $operation_name ] [ $operation_name.SERVICE.'Response' ] ,
                        WSDL_NAME,
                        WSDL_NAME."#".$server->wsdl->currentOperation,
                        "rpc",
                        "encoded",
                        $doc,
                        PATH_WSOAPENC_XS,
                        PATH_WSDL_NS."/in-out",
                        $ws_class_dic [ $server->wsdl->service_name ] [ SAWSDL_PARAMETER ]  );
 }
```

2.3.1.2 The multimodal companion

Thanks to the support provided by the Soa2m Semantic Services Library, the second step in the Soa2m project was the creation of a prototype to respond to the conference scenario and to test the advertisement of services and the serialization and deserialization of requests. The prototype allow us to test the instantiation of services. The test was produced by the implementation of two basic services, in a tool inspired on assistance services called the Multimedia Services Assistant.

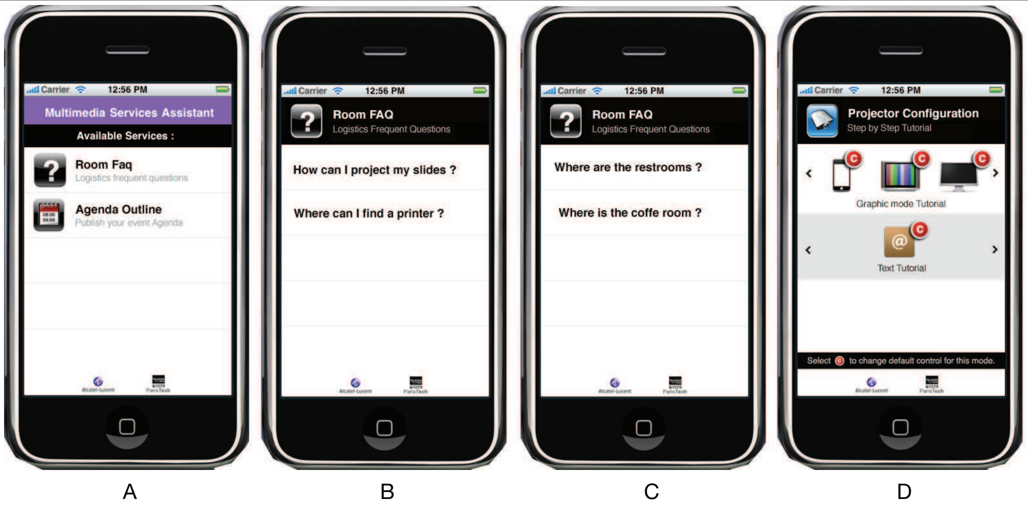

Multimedia Services Assistant
Available Services :
Room Faq
Logistics frequent questions
Agenda Outline
Publish your event Agenda
Room FAQ
Logistics Frequent Questions
How can I project my slides ?
Where can I find a printer ?
Where are the restrooms ?
Where is the coffe room ?
Projector Configuration
Step by Step Tutorial
Graphic mode Tutorial
Text Tutorial
Select to change default control for this mode.
A
B
C
D

Figure 2.3.4: Discovery of monomodal services in the Room FAQ

The design of the prototype is based on the proposition of a Room FAQ service and an Agenda Outline service for the conference [**Figure 2.3.4**] part A. The Room FAQ is designed to ask the more frequent questions about logistics related to the conference room and the Agenda Outline is designed to dynamically present the evolution of the conference in multiple modalities [**Figure 2.3.5**].

Goal of this test

The focus of this test prototype of companion wasn't the multimodal instantiation but the dynamic discovery of monomodal services in a local network and its instantiation in a test application. The idea was to test the services generation and parsing with the Soa2m Semantic Services Library and to prepare the mechanism for an implementation according with the MMI Framework & Architecture recommendation.

The services were discovered and instantiated according the situation (conference), the profile of the user (chair/attendee) and the state of the situation (in-conference/in-pause). For example, the list of Room FAQ services illustrated in [**Figure 2.3.4**] part B, is the instantiation of the discovered services available for a Chair, while in part C the same list present the services discovered for an attendee. If the chair selects the assistance service to project its slides, the service instantiates the tutorial in the best inferred modality and media (simulated in the prototype), and prevents the chair with the companion, to give him the possibility to choose another modality if the choice is not adapted to him. To provide these choices, the companion updates its register by a discovery process, filters the selection according to the situation (simulated in the prototype) and lists the other available options [**Figure 2.3.4**] part D.

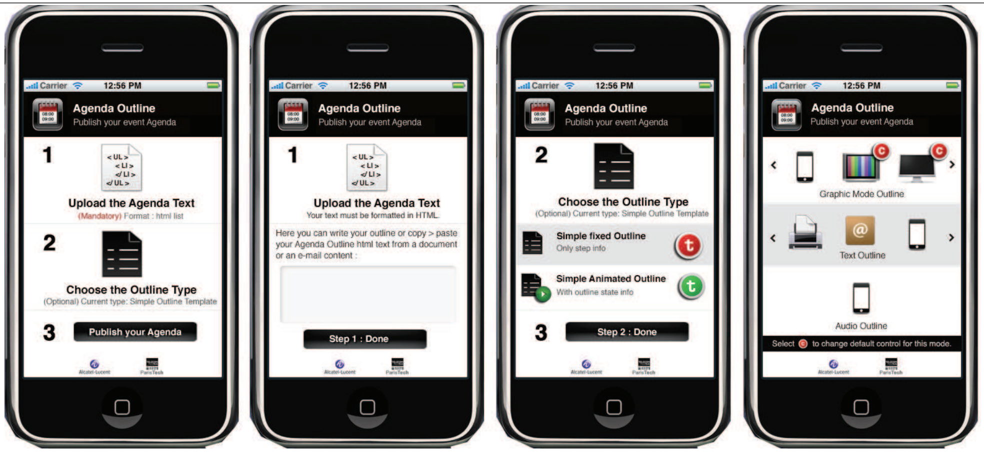
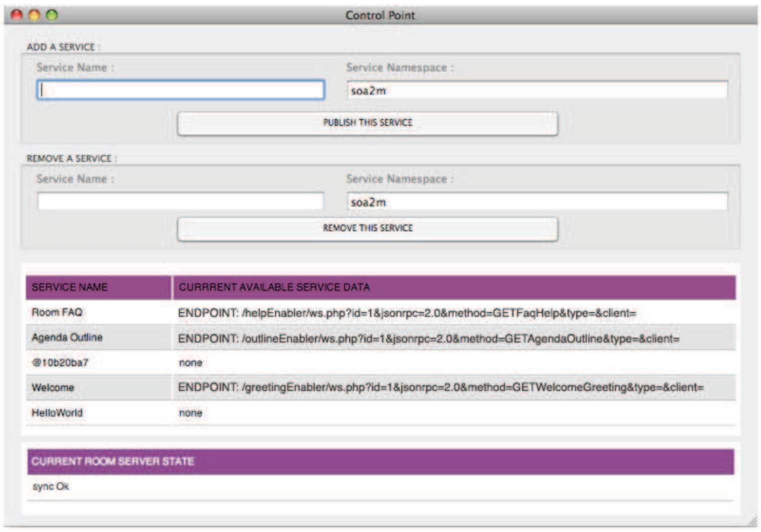

Carrier
12:56 PM
Agenda Outline
Publish your event Agenda
1
Upload the Agenda Text
(Mandatory) Format : html list
2
Choose the Outline Type
(Optional) Current type: Simple Outline Template
3
Publish your Agenda
Upload the Agenda Text
Your text must be formatted in HTML
Here you can write your outline or copy > paste your Agenda Outline html text from a document or an e-mail content :
Step 1 : Done
Simple fixed Outline
Only step info
Simple Animated Outline
With outline state info
Step 2 : Done
Graphic Mode Outline
Text Outline
Audio Outline
Select to change default control for this mode.

Figure 2.3.5: Discovery of monomodal services in the Agenda Outline

Finally, the discovery of the available services in the local network was ensured by a BonjourServicesListener MC service implemented to scan and discover the Soa2m service announcements [**Figure 2.3.6**].


Control Point

ADD A SERVICE :

Service Name :

Service Namespace : soa2m

PUBLISH THIS SERVICE

REMOVE A SERVICE :

Service Name :

Service Namespace : soa2m

REMOVE THIS SERVICE

SERVICE NAME	CURRRENT AVAILABLE SERVICE DATA
Room FAQ	ENDPOINT: /helpEnabler/ws.php?id=1&jsonrpc=2.0&method=GETFaqHelp&type=&client=
Agenda Outline	ENDPOINT: /outlineEnabler/ws.php?id=1&jsonrpc=2.0&method=GETAgendaOutline&type=&client=
@10b20ba7	none
Welcome	ENDPOINT: /greetingEnabler/ws.php?id=1&jsonrpc=2.0&method=GETWelcomeGreeting&type=&client=
HelloWorld	none

CURRENT ROOM SERVER STATE
sync Ok

Figure 2.3.6: Bonjour Discovery of services

To sum up, the implementation of this first prototype was the foundation for the design of the Soa2m Semantic Services Library and the creation of some services discovery tools.

2.3.2 *Scenario 2 - The mixed situation: Living Monuments.*

Imagine an old church where a series of events take place. Tourists can go in and out of the church before, after and during the services. During cultural events, believers can go in and out to pray. The church also have some social programs which means that also during charity events people in need or believers can be disturbed while tourist admire the monument.

The old church is a living monument, hosting simultaneous and mixed situations. The system discovers and instantiates the default services for each situation profile, according with the overall situation of the monument. Depending on the schedule and user position it can give access to different services. To believers, during services it can instantiate a Usher service, a Schedule service, a Guestbook service, a Bible service, a Karaoke service.

For the tourists it can instantiate the audio-guide service for the visit, composed according with the believers schedule in order to avoid the spaces reserved for pray. It can also instantiate a service explaining believers rituals in some reserved spaces in order to introduce some of the restrictions for the visit. It can also instantiate a souvenir service allowing the capture of sound, video, images and draws depending on the state of the church.

For the person in need, it can instantiate a route service to guide him to the places reserved to social events in an anonymous way (avoiding highly frequented zones), a Notification service for social events, an Alert service for extreme urgency needs.

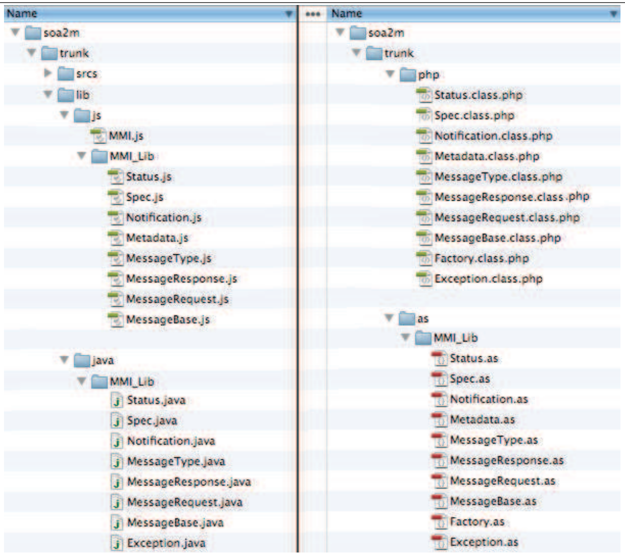
2.3.2.1 The MMI Implementation

Having already a tool to create and advertise modalities as semantic services, the next step in the Soa2m project was the implementation of an architecture allowing the multimodal coordination of services, as needed in the scenario above.

Goal of the implementation

With this goal, we launch the development of a library compliant with the MMI Architecture and Interfaces Recommendation, that at the time was a Working Draft lacking of test implementations.

The result is a set of libraries to support the generation[170] of the MMI Life-CycleEvents: the Soa2m MMILib [**Figure 2.3.7**]. Thus, the library is implemented for JavaScript, Java, Php and ActionsScript; opening the possibility of implementations with some of the more used programming languages for web development.



```
Name                                   Name
▼ soa2m                                ▼ soa2m
  ▼ trunk                                ▼ trunk
    ▶ srcs                                   ▼ php
    ▼ lib                                        Status.class.php
      ▼ js                                       Spec.class.php
          MMI.js                                 Notification.class.php
        ▼ MMI_Lib                                Metadata.class.php
            Status.js                            MessageType.class.php
            Spec.js                              MessageResponse.class.php
            Notification.js                      MessageRequest.class.php
            Metadata.js                          MessageBase.class.php
            MessageType.js                       Factory.class.php
            MessageResponse.js                   Exception.class.php
            MessageRequest.js
            MessageBase.js                   ▼ as
                                               ▼ MMI_Lib
      ▼ java                                       Status.as
        ▼ MMI_Lib                                  Spec.as
            Status.java                            Notification.as
            Spec.java                              Metadata.as
            Notification.java                      MessageType.as
            MessageType.java                       MessageResponse.as
            MessageResponse.java                   MessageRequest.as
            MessageRequest.java                    MessageBase.as
            MessageBase.java                       Factory.as
            Exception.java                         Exception.as
```

Figure 2.3.7: The four versions of the Soa2m MMILib

In the the Soa2m MMILib, the creation of a MMI event depends on a specification data structure (spec) that provides the information needed by an Event Factory to generate the MMI event markup according with the specification [**Figure 2.3.8**].

170 This work would not have been possible without the help and work of Hôa Nguyen.

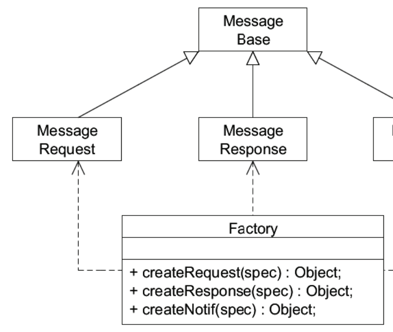

Message Base
Message Request
Message Response
Notification
Factory
+ createRequest(spec) : Object;
+ createResponse(spec) : Object;
+ createNotif(spec) : Object;

Figure 2.3.8: The Soa2m MMILib Factory

The following code illustrates the use of an MMI event in a Commander Modality Component. The user action (linked to a click in line 28) is handled by a javascript code «startIsClicked» in which the specification structure for the generation of the MMI event is described (line 4 to 14)). Then the API provided by the Soa2m MMILib is used to validate the spec and generate the markup (lines 17, 18, 19). Finally the event is sent in with a SOAP envelope to a semantic server (line 23), because the MMDispatcher component implements the Soa2m Semantic Services Library presented in section 2.3.1.1 The Web Services generation and annotation library.

```
1   <html>
2   <head><script type="text/javascript">
3     var startIsClicked = function() {
4           var spec = {
5                         method : orchestrate,       // config.js
6                         source : componentName,     // MC_config.js
7                         token : readCookie("token"),
8                         requestId : createRequestId(), // config.js
9                         target : serviceController,   // config.js
10                        metadata : MC_metadata,      // MC_config.js
11                        type : messageType.EXTENSION,
12                        confidential : MC_confidential, // MC_config.js
13                        context : readCookie("context"),
14                        status : status.SUCCESS
15          }

16          var mmiDispatcher = Object.create( MMDispatcher );
17          var myFactory = Object.create( factory ); //factory is defined in an import not included in this snipped
18          var msg;

19          try {  msg = myFactory.createRequest( spec );  }  catch  (err) {
20               alert("Exception: " + err);
21          }

22          // Spec OK. Send MMI Extension event with asynchronous SOAP request
23          mmiDispatcher.sendAsynchronous( msg,  "POST",  "MMI");
24     };
25   </script></head>

26   <body onload="init()">
27   <form name="form">
28         <input type="button" id="start" value="start" onclick="startIsClicked()" />
29   </form>
30   </body></html>
```

In this way, the creation and annotation of events and the requesting workload is significantly reduced, and again, a visual front-end can be produced to simplify the task of filling this structure.

Conclusion

To sum up, the Soa2m MMILib is the implementation of four APIs to generate MMI Life-Cycle Events and to implement its protocol.

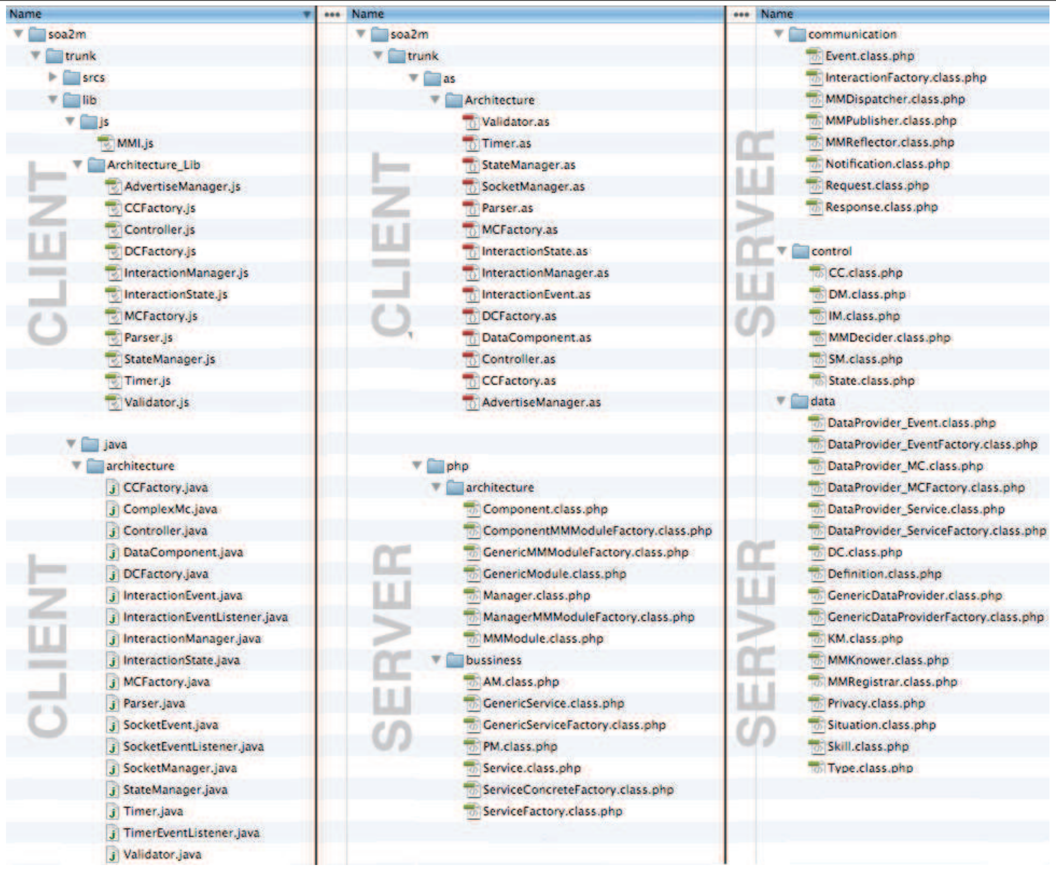
The library is compliant with the latest version of the MMI Architecture and Interfaces recommendation[171] and generates, validates and parses the MMI Events. It can be used with any multimodal architecture or multimodal system, to implement a communication protocol based on the synchronization of processes, which for us, was the main goal of our scenario. Thus, we create and use Soa2m MMILib library to handle the discovery, registering and update of Modality Component services in our semantic services oriented architecture.

2.3.2.2 The Soa2m library tool

Having an API to advertise and generate modalities as semantic services, an API to handle the discovery, registering and update of components ; and a communication protocol for multimodal events, the next logical step was the implementation of an adapted architecture capable to support the use of semantic services technologies for context-aware systems like the one described in our scenario. Then, we embrace the implementation of a library to create applications according with our proposed architecture[172].

The result is a library to support the creation of the Soa2m components and managers and its combination in the Soa2m architectural patterns. The library is implemented for the JavaScript, Java, and ActionsScript in the client-side Modality Components and in the server-side it is implemented in Php to be deployed in an Apache web server [**Figure 2.3.9**]. The library can be used with the previous libraries presented in this section, or without them.

Result: A library to implement the Soa2m architecture


```
Name
▼ soa2m
    ▼ trunk
        ▶ srcs
        ▼ lib
            ▼ js
                  MMI.js
              ▼ Architecture_Lib
                    AdvertiseManager.js
                    CCFactory.js
                    Controller.js
                    DCFactory.js
                    InteractionManager.js
                    InteractionState.js
                    MCFactory.js
                    Parser.js
                    StateManager.js
                    Timer.js
                    Validator.js

            ▼ java
              ▼ architecture
                    CCFactory.java
                    ComplexMc.java
                    Controller.java
                    DataComponent.java
                    DCFactory.java
                    InteractionEvent.java
                    InteractionEventListener.java
                    InteractionManager.java
                    InteractionState.java
                    MCFactory.java
                    Parser.java
                    SocketEvent.java
                    SocketEventListener.java
                    SocketManager.java
                    StateManager.java
                    Timer.java
                    TimerEventListener.java
                    Validator.java
```

```
Name
▼ soa2m
    ▼ trunk
        ▼ as
            ▼ Architecture
                  Validator.as
                  Timer.as
                  StateManager.as
                  SocketManager.as
                  Parser.as
                  MCFactory.as
                  InteractionState.as
                  InteractionManager.as
                  InteractionEvent.as
                  DCFactory.as
                  DataComponent.as
                  Controller.as
                  CCFactory.as
                  AdvertiseManager.as

        ▼ php
            ▼ architecture
                  Component.class.php
                  ComponentMMModuleFactory.class.php
                  GenericMMModuleFactory.class.php
                  GenericModule.class.php
                  Manager.class.php
                  ManagerMMModuleFactory.class.php
                  MMModule.class.php
            ▼ bussiness
                  AM.class.php
                  GenericService.class.php
                  GenericServiceFactory.class.php
                  PM.class.php
                  Service.class.php
                  ServiceConcreteFactory.class.php
                  ServiceFactory.class.php
```

```
Name
▼ communication
      Event.class.php
      InteractionFactory.class.php
      MMDispatcher.class.php
      MMPublisher.class.php
      MMReflector.class.php
      Notification.class.php
      Request.class.php
      Response.class.php

▼ control
      CC.class.php
      DM.class.php
      IM.class.php
      MMDecider.class.php
      SM.class.php
      State.class.php
▼ data
      DataProvider_Event.class.php
      DataProvider_EventFactory.class.php
      DataProvider_MC.class.php
      DataProvider_MCFactory.class.php
      DataProvider_Service.class.php
      DataProvider_ServiceFactory.class.php
      DC.class.php
      Definition.class.php
      GenericDataProvider.class.php
      GenericDataProviderFactory.class.php
      KM.class.php
      MMKnower.class.php
      MMRegistrar.class.php
      Privacy.class.php
      Situation.class.php
      Skill.class.php
      Type.class.php
```

Figure 2.3.9: The Soa2m Architecture Library

Thus, this library is a tool provided to allow the implementation of the Soa2m architecture.

---

171 The last version is the version published the 25 october 2012. For a complete specification of the MMILib See Appendix 2 Implementation Report

172 See supra section 2.1 The Soa2m Architecture.

For example, with the library it is possible to handle the availability and unavailability of components by putting them in unavailable mode during the events restricted in terms of space and scheduling in our scenario. This can be the situation for the audio-guide during religious services. This is achieved by using the adaptive pull mechanism, combined with the checkUpdate notification as defined in section 2.1.3.3.

Finally, to test the library and the protocol of communication, to activate and deactivate the interaction cycle[173] according with the situation processed by a decision engine, we implemented a series of monomodal services, based on the architecture's Smart Chronicler pattern.

These are atomic Modality Components with nested Controllers, called «complex modality components» in the MMI recommendation. These components are capable to advertise its services, to be registered and to communicate with MMI/Soa2m Events for being discovered, activated and deactivated by a controller component in addition of their default tasks:

	Taxonomic Family	Component Name	Description
Implemented Modality Components	**Commander**	htmlStatelessCommander	Controller for continuous media without handling of the reproduction state
		htmlStateCommander	Controller for continuous media with handling of the reproduction state
	Displayer	htmlFixedDisplayer	Displayer that can't be carried
		javaMobileDisplayer	Displayer that can be carried
	Listener	gpacServicesListener	Listener of UPnP services announcements
	Recognizer	flexSpeechRecognizer	Recognizer of Speech (english)
		flashGestureRecognizer	Recognizer of touch gestures
	Recorder	airVoiceRecorder	Recorder of voice sounds
	Synthesizer	flexRemoteSynthesizer	Synthesizer of streamed sounds (tested in english, french, spanish and vietnamese)
	Writer	flashGestureWriter	Writer of touch paths
		airTypeWriter	Writer of text strings
		flexTypeWriter	Writer of streamed text strings

Table 2.3.2: Modality Components Implemented

The detailed MMI implementation report of these components is available at:

http://www.w3.org/2002/mmi/2012/mmi-arch-ir/

The [**Figure 2.3.9**] illustrates one of the tests of modality communication and control realized for enabling disabling services based on contextual information. The flexGestureWriter MC captures gesture inputs, then the inputs is sent to the flashGestureRecognizer to interpret the semantics of the gesture and finally, if the controller decides that the context allows the execution of the command, the airVoiceRecorder starts its recording task.

Goal of this test

The implementation of this test series allow us to detect some open issues in the MMI Framework & Architecture Recommendation concerning issues like the exchange of continuous data combined with the control events between components and to explore some possible answers. This subject is proposed as a discussion and recommendation subject for the current charter of the Working Group.

173 For some implementation diagrams See Appendix 3 State Diagrams for registration

Then, the example [**Figure 2.3.10**] illustrates the distribution of remote modalities collaborating into one interaction cycle. That was the main preoccupation of the implementation of these series prototypes: to allow the test of communication with the MMI protocol of remote inputs and outputs and a Controller Component, handling a basic turn taking mechanism.

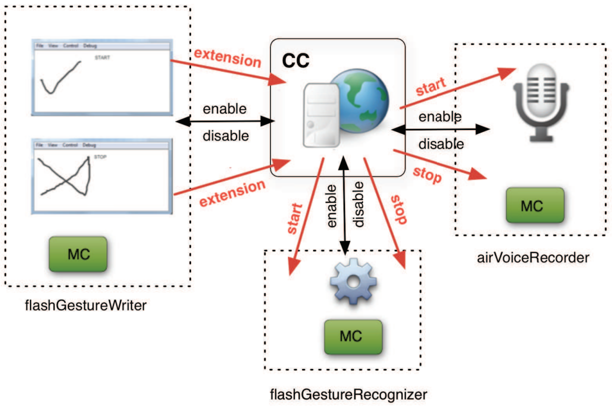


Figure 2.3.10: Test of enabling/disabling of modalities according to context

This turn taking, hard-coded in our tests, is the principal task related to the understanding of contextual situations. For this reason, our next concern was the enrichment of the architecture, in the server-side, with «intelligent» features implementing rules and behaviors from situation structures.

2.3.3 Scenario 3 - The Hypersensitive situation: The Elikya Center.

Imagine a center needing to integrate disabled children. The center want to propose children-oriented intuitive services to introduce them to rich media technologies in order to collaborate into their integration treatment. These children lived with any access to technology and are illiterate but they also are suspicious of adults. They can accept more easily to interact with a multimodal system than with a therapist and the center wish to use adapted multimodal services to support the integration process and physical and psychological therapy.

Clapping-based companions, a very articulated TTS component, drawing services or reduced gesture widgets can be some of the services instantiated according to the users needs and profiles. Other Modality Components can be proposed, like phone devices with very big symbols, very simple remote controls, screens displaying text at high resolution, voice command and haptic devices based on a reduced number or tasks.

2.3.3.1 A multimodal data model

After testing the communication protocol for discovery and registering of Modality Components, the need of context-handling as a tool for the selection of modalities was clear. For this reason, the Soa2m project focused in the production of implementations adapted to address the needs introduced by the scenario above. These results were intended to provide tools for the context management in *multimodal systems* enhanced with semantic technologies.

To begin, the scenario shows the need of a user model[174] adapted to multimodal situations. This means that the description of the user must focus on data near to the human body. This user model is the basis for the description of foundational multimodal notions like the *mode* and the *modality*.

174 See supra section 1.2.2.2 user Modeling

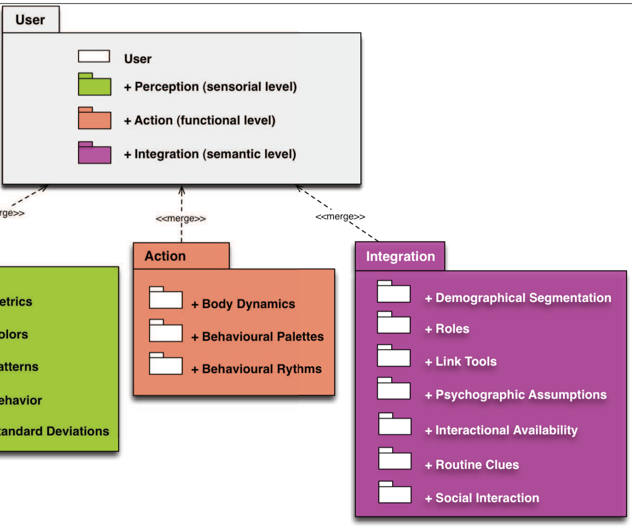


Figure 2.3.11: The Soa2m User Model

First, we proposed a user modell[175] [**Figure 2.3.11**] and second we implemented the basic ontologies needed for a Soa2m multimodal application.

Proposal: A user model

The Soa2m user mode is oriented to describe the multimodal interaction situation, and to describe agents in situations. [**Figure 2.3.12**] shows an example of the taxonomy of social situations needed for our scenario, to identify the patients in the community and in the adoption center.

It focuses in the premise used in Soa2m to structure all the knowledge: everything in context is a perdurant. The example shows how identity is related to a set of changing concepts, affected by social constraints and intentional facts. For example, some of the patients were social androgynes, given that sexual identity was seriously damaged, e.g. voice pitch not corresponding anymore to the classification male/female.

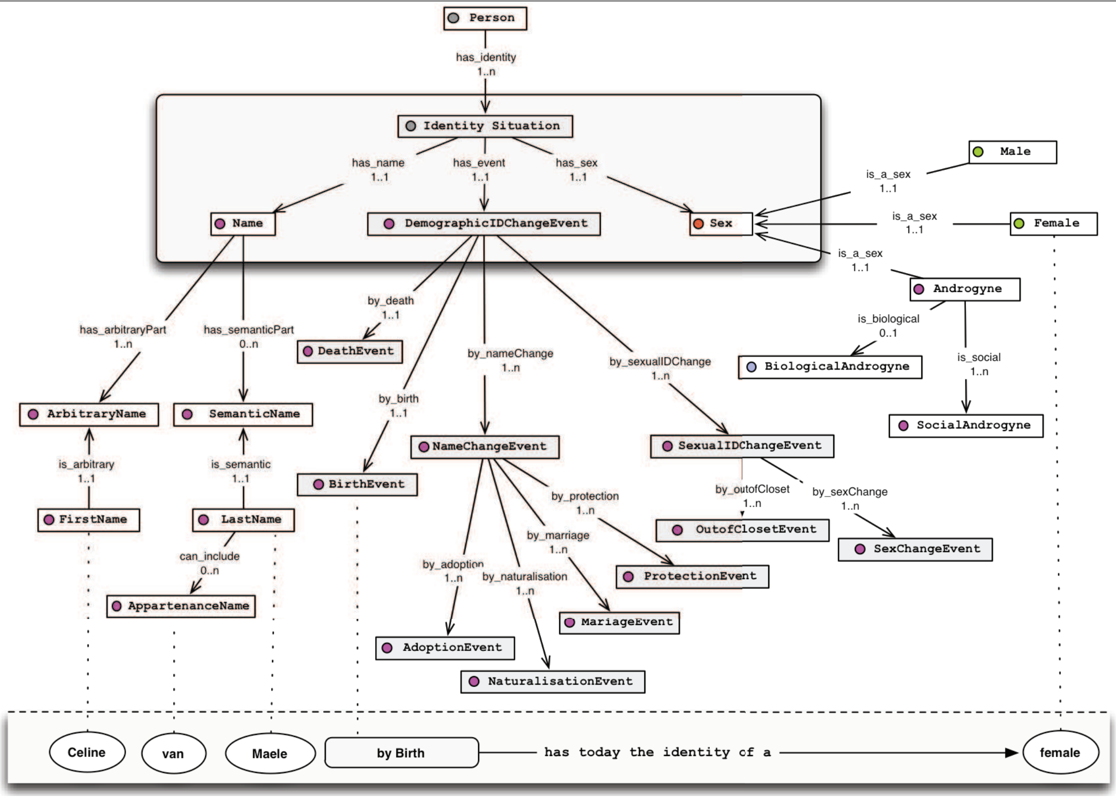
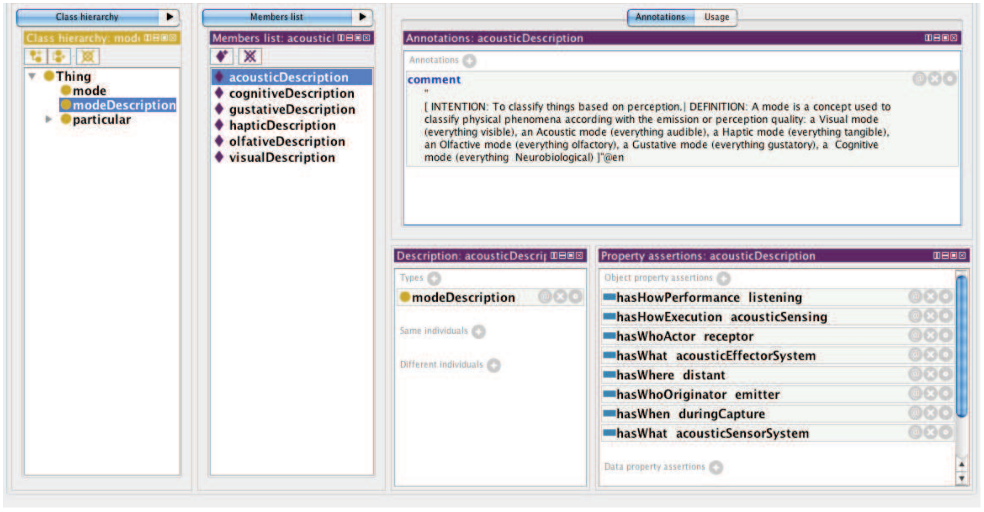


Figure 2.3.12: The User Model used in an Identity situation

175 For a more complete version of the user model See Appendix 4 User Model

This model is also the basis for the description of the *mode* as a perceptual phenomenon, which means, that the *mode* is a category of human sensorial perception. Thus in Soa2m we use the user model to describe the acoustic, visual, haptic, olfactive, gustative and cognitive modes. [**Figure 2.3.13**]



acousticDescription
cognitiveDescription
gustativeDescription
hapticDescription
olfativeDescription
visualDescription

comment
"
[INTENTION: To classify things based on perception.| DEFINITION: A mode is a concept used to classify physical phenomena according with the emission or perception quality: a Visual mode (everything visible), an Acoustic mode (everything audible), a Haptic mode (everything tangible), an Olfactive mode (everything olfactory), a Gustative mode (everything gustatory), a Cognitive mode (everything Neurobiological)]"@en

Types: modeDescription

hasHowPerformance listening
hasHowExecution acousticSensing
hasWhoActor receptor
hasWhat acousticEffectorSystem
hasWhere distant
hasWhoOriginator emitter
hasWhen duringCapture
hasWhat acousticSensorSystem

Figure 2.3.13: The Soa2m mode description

For example, to describe the acoustic mode in multimodal systems, we express the analogical relation between the «acousticSensing» (line 508) and the human hearing that needs at least one available ear (line 506) described in the Body Metrics category and some originator of the phenomenon creating sounds.

In the same way, we use the description of a ear as an analogical reference to describe an «acousticSensorSystem» (line 504).

And finally we can also relate the «duringCapture» property (line 512), with the Body Dynamics category that describes in the user model, the perception range.

In result, for example, the Soa2m ontology of acoustic modes was created, linking the sensor/effector behaviors with the human perception and the human body as described in the user model:

```
500  <owl:NamedIndividual rdf:about="&mode;acousticEffectorSystem">
501  <rdf:type rdf:resource="&mode;effectorSystem"/>
502      <analogically-references rdf:resource="&mode;vocalCords"/>
503    </owl:NamedIndividual>

504  <owl:NamedIndividual rdf:about="&mode;acousticSensorSystem">
505      <rdf:type rdf:resource="&mode;sensorSystem"/>
506      <analogically-references rdf:resource="&mode;ear"/>
507  </owl:NamedIndividual>

508  <owl:NamedIndividual rdf:about="&mode;acousticSensing">
509    <rdf:type rdf:resource="&how;executionValue"/>
510    <analogically-references rdf:resource="&mode;earing"/>
511  </owl:NamedIndividual>

512    <owl:NamedIndividual rdf:about="&mode;duringCapture">
513    <rdf:type rdf:resource="&DOLCE-Lite;time-interval"/>
514    <rdf:type rdf:resource="&which;whenValue"/>
515    <analogically-references rdf:resource="&mode;duringPerception"/>
516  </owl:NamedIndividual>

517    <owl:NamedIndividual rdf:about="&mode;duringPerception">
518      <rdf:type rdf:resource="&DOLCE-Lite;time-interval"/>
519      <rdf:type rdf:resource="&which;whenValue"/>
520    </owl:NamedIndividual>
```

```
<owl:NamedIndividual rdf:about="&mode;earing">
   <rdf:type rdf:resource="&how;executionValue"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&mode;ear">
   <rdf:type rdf:resource="&which;whatValue"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&mode;emitter">
   <rdf:type rdf:resource="&who;originatorValue"/>
</owl:NamedIndividual>

<owl:NamedIndividual rdf:about="&mode;receptor">
   <rdf:type rdf:resource="&who;actorValue"/>
</owl:NamedIndividual>
```

As we explained above, the User Model and the mode ontology can be reused for the annotation of situation and agents. For our scenario, this is a conceptual support in the description of disabled children in all the facets that can be pertinent to a multimodal interaction as a part of a recovering therapy. For example, the gesture recognition modality can be an important tool for an application, if the kid has his two hands.

If not, the system must be capable to avoid the instantiation of the service or its use in a multimodal interaction.

With the User Model as a starting point for the description of agents, the semantic description of situations became the next step to follow in order to enhance the Soa2m architecture with tools to handle the usage context.

2.3.3.2 The Soa2m situation ontology library

The last implementation provided by the Soa2m project is the ontology for the description of situations created according with the Soa2m situation model described in section 2.2.3.2 A Knowledge Model for Multimodal Discovery.

In multimodal interaction, three context ontologies to express the context of use are currently available. The NEXOF-RA Project[176] captures the context of use in which a user is interacting with a particular computing platform in a given physical environment in order to achieve an interactive task.

GUMO [Heckmann et al. 2005] is used to represent generic user models at a semantical level arranged in five aspects: mainpart , situation, explanation, privacy and administration. The user model dimensions are expressed into three parts: auxiliary, predicate and range. Finally, the Delivery Context Ontology (DCO)[177] provides a formal model of the characteristics of the environment in which devices interact with the applications.

Nevertheless, the granularity of all these ontologies is at the level of the task, and the situation is approached from the perspective of the human-system relation in a very detailed way.

For a more generic and social approach, we selected DOLCE. The ontology inherits of the concepts defined in DOLCE-Lite and the D&S Pluging, starting with the situation concept: the Soa2m situation is a DOLCE situation[178], that is not an agent (non-agentive social object) but a non-physical object described in DOLCE as an endurant.

[Figure 2.3.14] shows the alignments with DOLCE taxonomy of concepts.

176 NEXOF-RA Project, http://www.nexof-ra.eu

177 http://www.w3.org/TR/2009/WD-dcontology-20090616/

178 «A situation is a social object that appears in the domain of an ontology only because there is a description whose components can 'carve up' a view (setting) on that domain. A situation has to satisfy a description , and it has to be setting for at least one entity.» [Guarino et al. 2001]

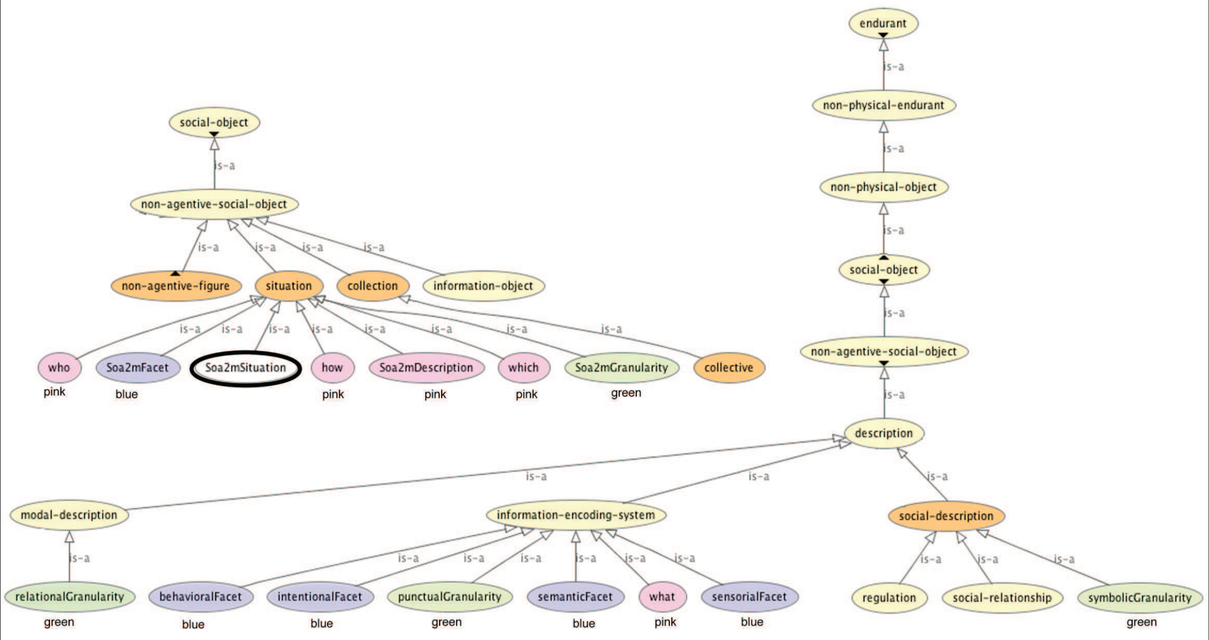


Figure 2.3.14: The Soa2m situation ontology aligned with DOLCE-Lite/DnS

The Soa2mFacet (in blue), the Soa2mGranularity (in green) and the Soa2mDescription (in pink) are also situations in the Soa2m situation ontology. [**Figure 2.3.14**] More precisely, these are situations of classification that an agent can define.

The elements of a Soa2mDescription (*who*, *how* and *which* -in pink) are also situations. For example, the activity necessary to interact with a multimodal system, can be an execution situation described corresponding to the sensorial facet of the *how* description.

The second tree in the graphic shows how the four Soa2mFacets (behavioral, intentional, semantic and sensorial) are descriptions arranged as information structures called in DOLCE «information-encoding-systems»[179].

The punctual granularity (in green), used to enumerate attributes of the situations; and the what description (in pink) are also information structures.

In contrast, the symbolic granularity (in green) is a social description, what means it is a product of social usages or agreements, sometimes not explicit.

Finally, the relational granularity is a «modal-description»,

A modal description describes a course, which is a concept that sequences perdurants (processes, events, or states), as a component of some description. In other words, its describes the phases of changing things.

While relations are changing phenomena, modal descriptions can divide in sequences the different aspects of the relation according to some temporal reference.

On the other hand, in the Soa2m ontology all the entities of the description are described by a pairs of data corresponding to the tuple <type,value>.

For example, agents are described by a <role,endurant> pair.

In this way, the *who* attribute actor is a role, that can only be played by agents. [**Figure 2.3.15**] shows what an agent can be in the DOLCE taxonomy: a rational agent (a person[180]) or an agentive physical object (a smart system[181]) or an agentive social object (a government[182]).

179 «An information encoding system is a description that involves information objects. They can be divided into 1) axiomatic systems, which provide roles and operations to define formal descriptions (e.g. theories), 2) combinatorial systems, which provide roles and operations to create valid information objects (e.g. grammars), 3) classification systems, which are contexts of (ev. ordered) lists of information objects, and 4) informal encoding systems, which provide roles and operations to define informal descriptions (e.g. narratives).» [Guarino et al. 2001]

180 Definition that we can align as based in a punctualGranularity.

181 Definition that we can align as based in a relationalGranularity.

182 Definition that we can align as based in a symbolicGranularity.

By extension, in Soa2m the actorValue, originatorValue and inspiratorValue are of these three kind of agents.

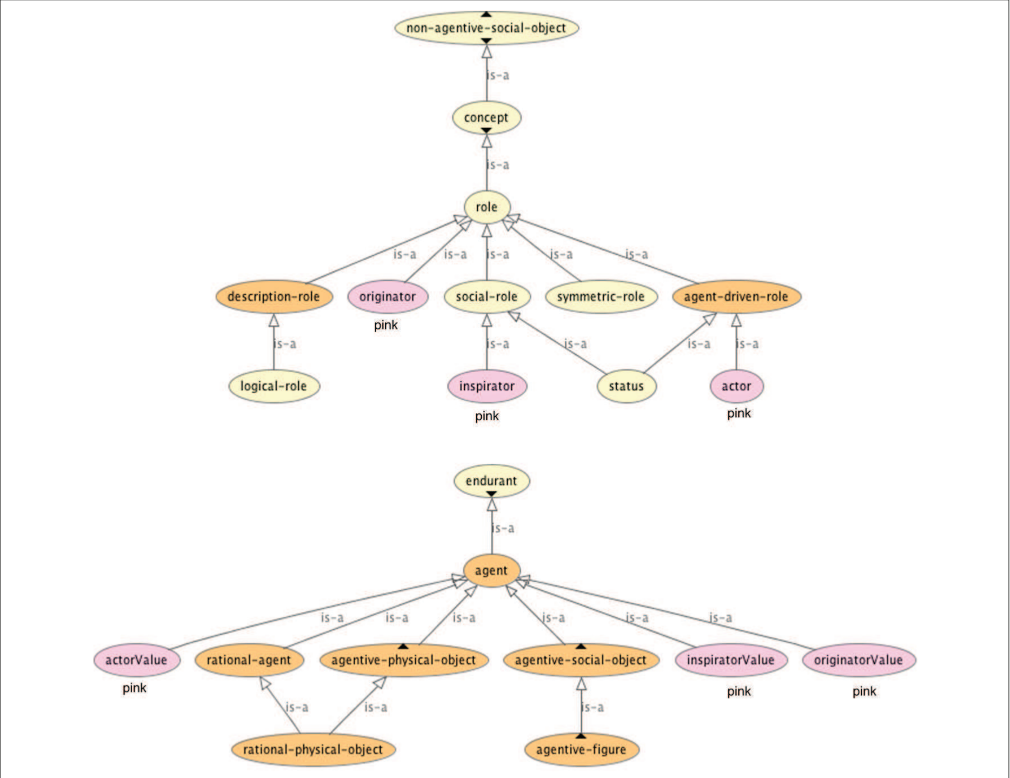


Figure 2.3.15: The Soa2m situation Who aligned with DOLCE-Lite/DnS

Another example of the use of <type,value> tuples is the description of the Soa2mDescription attribute *how*. [**Figure 2.3.16**] As we explained in section 2.2.3.2 A Knowledge Model for Multimodal Discovery, the *How* description it is composed of three complementary informations depending on the granularity of the description: execution, performance and intention.

In the **punctual granularity**, the attribute is called execution, in the **relational granularity**, the attribute is called performance and in the **symbolic granularity**, the attribute is called intention.

To annotate any of these three attributes, we need to provide a tuple <type,value>. According with the Soa2m situation knowledge model, the type of any *How* value MUST be DOLCE methods (in pink) [**Figure 2.3.16**]: the performance corresponds to a practice, the intention to a plan and the execution to a technique. In other words, to describe a situation from a *How* perspective we can only express methods: e.g. the type of an execution is any information corresponding to a method. When annotating any given situation, this value is restricted to a method «dataType».

On the other hand, for two of them, the values of the attributes are perdurants: the performanceValue is an activity while the intentionValue is an achievement[183]. For the other, the executionValue, is an endurant, corresponding to the task [184]concept. In other words, to describe a situation from a *How* perspective the values of the methods are of different categories: e.g. the value of an execution is a task (which is a specific type of method).

183 «Eventive occurrences (events) are called achievements if they are atomic, otherwise they are accomplishments.Further developments: being 'achievement', 'accomplishment', 'state', 'event', etc. can be also considered 'aspects' of processes or of parts of them.» [Guarino et al. 2001]

184 A task is "used to sequence activities or other controllable perdurants (some states, processes), usually within methods. (…) Tasks can be complex, and ordered according to an abstract succession relation. A task is different both from a flowchart node, and from an action or action type.Tasks can be considered shortcuts for plans, since at least one role played by an agent has a desire attitude towards them (possibly different from the one that puts the task into action). In principle, tasks could be transformed into explicit plans."[Guarino et al. 2001]

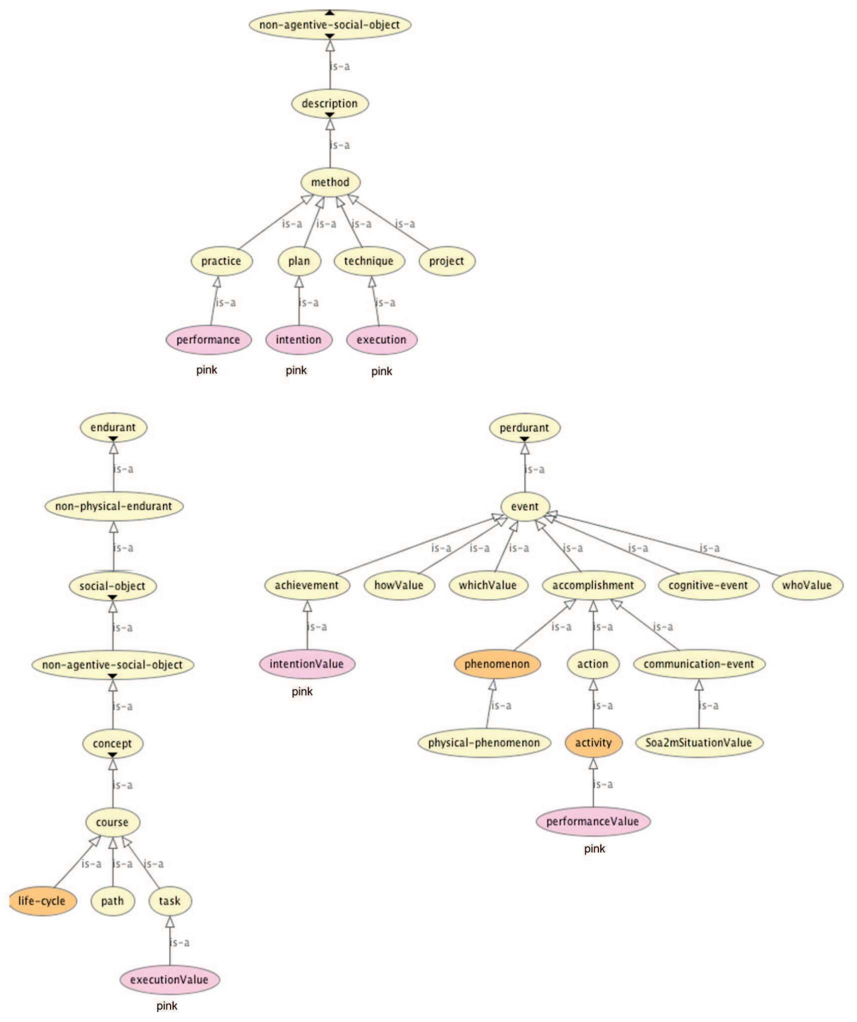


Figure 2.3.16: The Soa2m situation How aligned with DOLCE-Lite/DnS

In result, a situation *How* described in a punctual granularity will provide information formatted according with the tuple <method,task>. An inference engine will suppose that any data provided to describe this aspect of the situation will correspond to this high level «dataTypes».

For example, the following instance of a Soa2m situation (a NamedIndividual) describes an alerter modality in a semantic facet. In this description, lines *755* to *757* correspond to the description of the *How* aspect. We know by the presentation of axioms above, that the type of the element in line *756* <Soa2mSituation:**hasHowExecution**> in Soa2m is always a method and its value (the resource reference) is always a task. This corresponds to the tuple <type,value> of every concept in the data model of our design pattern. The tuples are instantiated here by using the tag elements of the Soa2mSituation namespace.

```
750  <owl:NamedIndividual rdf:about="&modality;alerterDescription">
751    <rdf:type rdf:resource="&modality;modalityDescription"/>
752   <Soa2mSituation:hasWhoActor  rdf:resource="&notiffier;receivers"/>
753    <Soa2mSituation:hasWhoOriginator  rdf:resource="&notiffier;sender"/>
754    <Soa2mSituation:hasWhoInspirator rdf:resource="&notiffier;reporter"/>

755    <Soa2mSituation:hasHowExecution  rdf:resource="&notification;accousticNotification"/>
756    <Soa2mSituation:hasHowPerformance  rdf:resource="&emotion;affraid"/>
757    <Soa2mSituation:hasHowIntention  rdf:resource="&goal;alert"/>

758    <Soa2mSituation:hasWhere  rdf:resource="&zone;intimate"/>
759    <Soa2mSituation:hasWhen  rdf:resource="&delay;immediate"/>
760   <Soa2mSituation:hasWhat  rdf:resource="&notiffier;message"/>
761  </owl:NamedIndividual>
```

In conclusion[185], the Soa2m situation pattern is a tool to annotate knowledge about the context of multimodal interaction. This is an ontology, allowing the semantic description of situations, for an IA-based decision engine. It was developed with Protegé and tested with the Pellet reasoner engine.

As we show in with the examples presented in section 2.2.2.3 Services Description with WSDL Documents, the ontology can be used to classify tags or to align other ontologies, for example, the service description manifests. It can be also used for the semantic discovery and registering of services when used with WSDL and OWL-S.

2.3.4 Conclusion

In this section we presented a list of the implementations produced in Soa2m as a result of the analysis of generic requirements in three hypothetical situations, under the light of the task «services discovery and registering».

Three types of technical requirements were detected:

- the need to semantically annotate web services to take advantage of the web of knowledge in *fusion*, *fission* and context handling of multimodal systems.
- the need of an adapted architecture and a standardized protocol of communication to enable, disable and coordinate distributed services in multimodal systems.
- the need of an adapted semantic model for multimodality centered in the user perception, behavior and social integration.

To these requirements we respond with five implementations of libraries as a contribution to the adoption of multimodal standards:

- The Soa2m Semantic Services Library as a tool for the annotation of Modality Component services with semantic web technologies
- the Soa2m MMI Lib as a tool for the implementation of the MMI protocol of communication.
- the Soa2m Architecture library as a tool for the implementation of context-aware and semantically intelligent architectures extending the MMI Architecture and Interfaces recommendation.
- the Soa2m Situation ontology to enrich multimodal systems with context knowledge.
- the user model and the mode ontology to describe multimodal phenomena.

For a timing reason, these tools were internally validated with some prototypes and tests that probed their functioning and integration but a complete validation in a global application is still needed. Nevertheless, a validation was received from the official paths of discussion and evaluation of standardized implementations at the W3C.

As a contribution to an open standard, we expect that feedback will come of potential users during the life-cycle of the targeted recommendations. Nevertheless, the MMI Architecture recommendation become an official open standard in October 25 / 2012. This implies that the validation and evaluation by users is a process to come.

185 To consult a synthetic list of the axioms of the ontology See Appendix 5 The Soa2mSituation

III The contribution

One of the initial goals of multimodal interaction with systems, is to enhance naturalness in the communicative exchanges with humans. One of the means to fulfill this goal is to allow machines to achieve human-level intelligence. By this, the way humans interact with them is supposed to change.

According with some current trends in Artificial Intelligence described in Chapter I - Multimodal Systems, we will have to stop treating machines as machines and as developers, we need to start programming machines, and transmit to them our knowledge, as if they were social entities like us.

Although there is a lot of work to do to describe experiential cognitive structures for a multimodal system, from a practical perspective a lot has already been accomplished.

Multimodal inputs and outputs in current systems has achieved a notorious maturity, in systems like Siri or touch applications, or in speech, sound and image recognition; and sensorial experience of multimodal events is rich. The data from different modes is detected and stored, and interaction and activities are tracked to be further used by statistical learning processes. However, all the information integration and instantiation lacks of intelligent context handling based on well grounded common sense notions.

We believe that this kind of semantics will enrich the autonomous behavior of multimodal applications, allowing robust perception, interaction control and semantic integration. But we have to start by the beginning. And for large scale systems, the beginning is the annotation of knowledge, its discovery and its registering.

3.1 Motivation.

The motivation behind this thesis work is the proposal of discovery and registering mechanisms in a services oriented architecture fully focused on multimodality and enhanced with semantic technologies. However, the technology necessary to achieve such a goal is not yet available in an standardized way, both at the services and the semantic levels. What is true for multimodal system architectures, namely, the lack of standardized design rules that could lead to a Theory and a more Automatic application of principles; is also true for the semantic web services.

This lack of technology will probably disappear in the next few years and this thesis aims at developing a core set of open standard contributions towards starting to solve this limitation.

With this goal, we draw inspiration from the performing arts, and more precisely from the experience achieved with *situated* interactive multi-sensorial installations to reach the creation of some tools to build (using the resources provided by the cloud) the next generation of intelligent and ludic web-based multimodal experiences, instead of just industrial automations.

3.2 Main contributions.

The main contributions of this thesis to the field of multimodal semantic services architectures are separated into conceptual and technological ones.

3.2.1 The conceptual contribution.

The goal of creating rich experiences and multimodal situations requires imaginative ways to formulate and approach research problems. This is a very interdisciplinary area. Human Computer Interaction, Cognitive and Neurosciences, as well as Philosophy, Ethnography, and Developmental Psychology, are good fields to look for inspiration. But Performing Arts are much better even if extrapolating lessons from nature and art to a multimodal system is a hard scientific problem.

In the Soa2m project we tried to push for new concepts to discover what it takes to build an intelligent, embodied and situated multimodal system.

More specifically, the conceptual contributions of this thesis include new approaches to tackle research issues using new original assemblies of multiple knowledges (description logics, multimodality, performing arts, business services). Thus, we list as conceptual innovation the large scope of applications covered by this high-level approach (at least for the semantic web, multimodality and SOA architectures) and the large range of problems it can solve.

An Embodied and Situated interpretation of multimodal interaction

Embodied and situated perception consists of boosting the capabilities of an artifact by fully exploiting the concepts of an agent with a body (or an analogical body) situated in the real world. These artifacts exist in dynamic environments that are manipulated or changed through their actions and are sensed or perceived by them. Thus, intelligent behavior derives from the environment, and the interactions are defined by the agent's embodiment.

Soa2m embraces in a coherent and persistent view this approach, by using: personification, situated descriptions directly related with the human perception of the world, the use of analogy as a description tool, and the proposal of an user model adapted to the current *multimodal system* needs.

In Soa2m, multimodal interaction is not a phenomenon to describe, followed by a situation that gives contextual information. In contrast, a situation is the previous condition of existence for any phenomenon, event or thing, and the most important condition for its description. The totality of the Soa2m proposal is builded around this conceptual point of view: Things exists (and can be described) only in situations.

The description facets

Structural relations between information in a situation can be a matter of subjectivity and interpretation. The description facets support such different interpretations of the same situation.

The description of a situation depends on the context of the observer and the interests of the receiver of the description. The same situation might have differing and complementary descriptions depending on the identification and clear separation of their different aspects, their separation of concerns.

Thus, a situation is the superposition of four facets of descriptions, each facet corresponding to a layer of the conceptual model of the Soa2m architecture: sensorial, behavioral, semantic and intentional.

These facets are reflected not only in the Soa2m situation, but also in the three methods for the annotation of services proposed in Soa2m, the extensions for the MMI Framework & Architecture and the Soa2m architectural design patterns.

While web services descriptions are currently supporting in a formalized way the modeling of the negotiation of interests for the services exchanges, the multimodal research address this problem from an opportunistic perspective, mostly related to the *fusion* process. Thus our first conceptual contribution points to this research subject and proposes a description approach recovering the advances in web services annotation and instantiation, that could help not only to the discovery of components in a multimodal system but also to the coordination and interpretation of the information they provide.

The granular view of the multimodal interaction

The Soa2m granularities are the conceptual application in the description of situations of three well-known orientations coming from the philosophy of science: empiricism, phenomenalism and hermeneutics. These orientations refer to world views and ways of thinking for understanding the nature of knowledge and reality, and are reference frames for a thorough analysis of the issues addressed in the contemporary cognitive sciences and artificial intelligence for data management and intelligent systems design.

Empiricism is a theory of knowledge that asserts that knowledge comes only or primarily from sensory experience: empirical data are data produced by an observation and can be analyzed quantitatively or qualitatively. This is the origin of the punctual granularity and implies enumeration and absolute measurement of the observations described in a situation seen as a sets of objects acting and reacting upon one another.

Phenomenalism affirms that statements (or descriptions) about physical objects are synonymous with statements about persons having certain sensations or sense-data. Physical objects cannot justifiably be said to exist in themselves, but only as perceptual phenomena or sensory stimuli situated in time and in space (perdurants). This is the origin of the relational granularity and implies the description of relations, phenomena, perceptions, functions and temporal intervals and semi-intervals.

Hermeneutics, broadly, is the art and science of interpretation, encompassing written, verbal, and nonverbal communication. In this approach, communication contents are conventionalized expressions of the experience of the author; thus, the interpretation of such contents will reveal something about the social context in which they were formed, but, more significantly, provide the receptor with a means to share the experiences of the author, and to perceive as the interpreter perceives these experiences. This is the origin of the symbolic granularity and implies the view of the situation as the performance of a scenario (musical, visual or any other plan) and as the interpretation of the situation according with an agent's (the interpreter) perspective.

Thus, the Soa2m situation is a holistic contribution to the semantic description of context, in which each granularity is a layer composed of three aspects (open sets) going beyond the description of attributes situated in time and space. This contribution inspired on the multimodal field enriches the contextual annotation of services and allows the user-oriented discovery and selection of distributed services.

3.2.2 The technological contribution.

This thesis extends and completes an open standard and explores new research problems in the field of distributed multimodal interfaces implemented with web technologies. As stated hereafter, some technological contributions consists of incremental improvements, while other propose a new approach.

Semantic Annotation of Services

In Soa2m we contribute to the annotation of services not only with semantics related to their processes but also with semantics related to the situation of use described from a common sense perspective. Currently (2012) none of the tools available to generate service descriptions is adapted to the latest standardization proposals to enhance WSDL with semantics to increase automation and interoperability.

We showed how domain knowledge stored in ontologies can be used with standard operations research techniques for process configuration and optimization. Specifically, we presented a multi-paradigm constraint analysis approach (the Soa2m situation) for allowing reasoning over logical and qualitative constraints. A tool to annotate, parse and instantiate these kind of services is proposed as a technical contribution of Soa2m. The tool allows integrating configuration and adaptation capabilities into standard Web process engines by adding description references to services descriptions, specifying the association between the WSDL or XML Schema component and a concept in some semantic model.

We believe that this tool will be useful for both business services and multimodal processes. One of the biggest challenges facing service providing is to have the ability to optimally configure their global workforce and provide new combined services; in our case, new compositions of multimodal user interfaces based on distributed widgets and recognition/synthesis processes.

Semantic State handling for multimodal systems

In Soa2m we contribute to the management of the state of components and the overall state of the multimodal system by using semantic annotations with two synchronization mechanisms (a push and an adaptive pull), with the use of two new events (checkUpdate event and UIUpdate notification) and with the proposal of some adapted architectural building blocks.

These resources combined allow the components to transmit the knowledge needed to initially register in detail their capacities and their use, but also to differentially update the changing information in this description.

Modality components with fast state changing (mouse, joystick, eye-tracker, gesture recognizer) can use the data component where recent states are saved and can use the update mechanism to control and send streams of data with periodical referential descriptions used to define the most important changes in these data. Thus, the two mechanisms used to transfer state changes in Soa2m are also a contribution in the direction of the stream handling and real time processing with the MMI Framework and Architecture.

The multimodal situation ontology pattern

Computational ontologies encode a description of some world for some purpose. They have a (primarily logical) structure, and must match both domain and task: they allow the description of entities whose attributes and relations are of concern because of their relevance in a domain for some purpose, which in our case is modality discovery, registration, query, search, integration and matching for context-aware multimodal systems.

Under the assumption that there exist classes of problems that can be solved by applying common solutions we propose a situation ontology to support reusability on the design of this kind of multimodal systems.

Thus, the Soa2m situation ontology is our contribution to the holistic description of the context of use of modality components treated as services in a multimodal system. The ontology is designed with a contextualised cognitive perspective. For example, we need to detect the attendance rate of a room to define the output sound volume for a given multimodal application.

Then a search engine should be able to retrieve all the sensors that are sensing the attendees presence in this specific room. This is possible only if sensors themselves expose information about what they are sensing. The question is: how can they understand automatically what they are sensing?

While humans understand a situation better when it can be associated with a similar past experience stored in memory, the same mechanism can be used to let a modality component service (wrapping a sensor) understand a situation. Then, technologically, modality components can emulate this human cognitive and associative mechanism by searching for similar situation from the past, using the multimodal system registry or the semantic web. The Soa2m situation ontology is a tool to improve and enable its understanding of what is happening around it (real world) and of what it is actually sensing (self-awareness of the modality component).

The Soa2m situation ontology is specific enough to be used automatically or semi-automatically, and at the same time generic enough to be useful in several ontologies for multimodal and context-aware systems. This is a pattern that is generic and domain independent, to support the manual construction of application and domain ontologies, and to contribute to the annotation of non-functional conditions affecting the multimodal system global state and the specific components behavior.

3.2.2 The contribution to the multimodal standardization.

A special contribution of this thesis concerns the standardization of multimodal architectures. We collaborate on the specification of the MMI Architecture and Interfaces in the following aspects:

- A complete implementation of the MMI Architecture and Interfaces (with 7 complex modality components and 1 global interaction manager) analyzed and described by an extensive implementation report. This implementation has been used as one of the four implementations that allow to validate the standard.
- The coordination and leadership of the new Discovery and Registration subgroup of the Multimodal Interaction Activity, which leads to the proposal of a set of use cases and their requirements. Three of the use cases, and the set of requirements were originated on the Soa2m proposal and validated by the W3C recommendation process.
- The proposal of four specific contributions for the next open standard to be published by the MMI Working Group: two architectural extensions in order to support state handling, two new events to support discovery management requirements, two new protocols for modality discovery, a modality component announcement for bootstrapping; and the creation of a common and interoperable vocabulary of generic skills to allow the gross discovery of modalities in large-networks over the concrete networking layer.
- The promotion and documentation of the MMI Architecture in multiple languages through the internet and the social media, like Wikipedia.

3.3 Perspectives.

This thesis placed special emphasis on the discovery and registering of multimodal services. This was motivated by the fact that for some of these problems, previous solutions were not distributed and off-line. Hence, the removal of this off-line requirement led to the development of on-line and largely distributed approaches for which discovery, selection and registering become a real issue.

To facilitate the description of the context of use in multimodal systems, a series of situation templates could be very important, by the detection of some of the more common use cases. These templates can be distributed in a common library, or by the construction of an authoring tool. This is a work that remains to be done.

On the other hand, for the XML-based discovery processes, it is mandatory to optimize the data load exchanged over the network. This can be explored by the proposal of an update mechanism based on differential descriptions. This is a work that is only in a starting state.

Finally, a common and interoperable vocabulary for multimodality is also a work to be done. From the state of the art we could see the different perspectives and means that the same multimodal notions can have (sometimes they are even contradictory).

A common vocabulary to be used by applications and systems will enhance interoperability in the semantic web of distributed multimodal widgets or services. Languages like EMMA lacks of a well-defined data model. A work to be done is the semantic definition and alignment of its concepts and multimodal knowledge with other well-known and very complete proposals like the Open Interface, the current work of the ARIA working group, the Web Intents joint task force or the Model-Based UI Working Group at the W3C. The participation on these groups is a motivating perspective while the production of a complete implementation in a prototype will be our next challenge.

3.4 List of Publications and other contributions

The following is a chronological list of our contributions and publications:

RODRIGUEZ,B.H. (Ed). **BARNETT**, J., **DAHL**, D., **TUMULURI**, R., **WIECHNO**, P. and **ASHIMURA**, K. Registration & Discovery of Multimodal Modality Components in Multimodal Systems. W3C Working Draft. To be published in March 2013.

BARNETT, J. (Ed). **BODELL**,M., **DAHL**, D., **KLICHE**,I., **LARSON**,J., **PORTER**, B., **RAGGETT**, D., **RAMAN**, T.V., **RODRIGUEZ**,B.H., **SELVARAJ**,M., **TUMULURI**, R., **WAHBE**, A., **WIECHNO**, P. and **YUDKOWSKY**, M. Multimodal Architecture and Interfaces. W3C Recommendation 25 October 2012. Available at: http://www.w3.org/TR/mmi-arch/

PETER,C., KARPOUZIS, K., and BIEBER, G. (Ed). "Multimodal Interfaces For Pervasive Assistance". A special issue of the Springer Journal on Multimodal User Interfaces. To be published in December 2012. Member of the Editorial Committee.

RODRIGUEZ,B.H. (Ed). **DAHL, D., TUMULURI**, R., **WIECHNO**, P. and **ASHIMURA**, K. Registration & Discovery of Multimodal Modality Components in Multimodal Systems: Use Cases and Requirements. W3C Working Group Note 5 July 2012. Available at: http://www.w3.org/TR/mmi-discovery/

DAHL, D., LARSON,J., RODRIGUEZ,B.H. and SELVARAJ,M. Best practices for creating MMI Modality Components. W3C Working Group Note 1 March 2011. Available at: http://www.w3.org/TR/mmi-mcbp/

RODRIGUEZ,B.H. and **MOISSINAC,** J.C. “Semantique pour les capteurs et effecteurs en environement pervasif” in Proceedings of the 7 èmes Journées Francophones Mobilité et Ubiquité UBIMOB 2011. Toulouse, France. 6-9 juin 2011.

RODRIGUEZ,B.H.**, MOISSINAC,** J.C. and **DEMEURE**, I. “Multimodal Instantiation of Assistance Services” in Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services. MOMM10-iiWAS 2010. Paris, France. 8-10 november 2010.

RODRIGUEZ,B.H.**, MOISSINAC,** J.C. **DEMEURE**, I. and **DUPONT**, M.P. “Multimodal services for the pervasive semantic web” in Proceedings of the 6 èmes Journées Francophones Mobilité et Ubiquité UBIMOB 2010. Lyon, France. 7-9 juin 2010.

RODRIGUEZ,B.H.**, MOISSINAC,** J.C. and **DEMEURE**, I. “A semantic model for multimedia services composition” in Proceedings of the Summer School on Multimedia Semantics, Koblenz, Germany, 23.-28. August 2009

RODRIGUEZ,B.H.**, MOISSINAC,** J.C. and **DEMEURE**, I. “« User experience semantic models for multimedia services composition in pervasive systems”, Intermedia Summer School 2009, Chania, Crrete, Grèce.

RODRIGUEZ,B.H. "Multimodal Architecture and Interfaces". Wikipedia Page: Available at: http://en.wikipedia.org/wiki/Multimodal_Architecture_and_Interfaces and http://fr.wikipedia.org/wiki/Architecture_Multimodale_et_Interfaces

REFERENCES

1 Books, Articles and Unpublished Material.

[Abel et al. 2010] **ABEL**, F., **HENZE**, N., **HERDER**, E., **HOUBEN**, G., **KRAUSE**, D. and **LEONARDI**,E. "Building Blocks for the User Modeling with Data from the Social Web" In: *Proceedings of the International Workshop Architectures and Building Blocks of Web-Based User-Adaptive Systems (WABBWUAS 2010) at UMAP 2010*. Hawaii, USA. Available (04/02/2012) at: http://www.l3s.de/web/upload/documents/1/paper7.pdf

{Adjie-Winoto et al. 1999} **ADJIE-WINOTO**, W., **SCHWARTZ**,E., **BALAKRISHNAN**,H. and LILLEY,J. "The Design and Implementation of an Intentional Naming System". Proc. 17th ACM SOSP, Kiawah Island, SC. Dec. 1999.

[ADOBE-Flash 2012] Technical information at: http://www.adobe.com/devnet/flashplatform/whitepapers/roadmap.html

[Akman et al. 1996] **AKMAN** ,V. and **SURAV**, M. "Steps toward formalizing context.» In: *AI Magazine*, 1996, Vol. 17. No.3. pp. 55-72.

[Allen et al. 1985] **ALLEN**, J.F. and **HAYES**, P.J. «A Common-Sense Theory of Time: The Longer Paper» TR Computer Science Dept, Univ. of Rochester 1985.

[APPLE-AppleTalk 1985] **SIDHU**,G., **ANDREWS**,R. and **OPPENHEINER**,A. "Inside AppleTalk, Second Edition", Addison-Wesley, 1989, ISBM 0-201-55021

[Arsenio 2004] **ARSENIO**, A. M. "Towards an Embodied and Situated AI" In: *Proceedings of the International FLAIRS conference*, 2004. Available (04/02/2012) at: http://www.aaai.org/Papers/FLAIRS/2004/Flairs04-124.pdf

[Atrey et al. 2006] **ATREY**, P., **HOSSAIN**, M., **EL SADDIK**, A. and **KANKANHALLI**, M. "Multimodal fusion for multimedia analysis: a survey" In: *Multimedia Systems*, Nov. 2010. Springer, Berlin. Vol. 16. No. 6. pp. 345-379 http://dx.doi.org/10.1007/s00530-010-0182-0

[Austin 1962] **AUSTIN**, J. L. How To Do Things With Words. Oxford University Press, 1976. (Second Edition). 40 pages.

[Balbiani et al. 1998] **BALBIANI**, P., **CONDOTTA**, J-F and **FARIÑAS DEL CERRO**, L. "A model for reasoning about bidimensional temporal relations" In: *Proc. of the International Conference on Principles of Knowledge Representation and Reasoning* KR'98. Morgan Kaufmann, 1998. Available (04/02/2012) at: http://www.irit.fr/ACTIVITES/LILaC/Pers/Condotta/publications/kr98.ps.gz

[Barraquand et al. 2010] **BARRAQUAND**, R. and **CROWLEY**, J. "Acquiring Common Sense Knowledge From Smart Environments" in *Proceedings of AAAI Fall Symposium on Commonsense Knowledge*, Arlington, Virginia, 2010. Available (04/02/2012) at: http://www.aaai.org/ocs/index.php/FSS/FSS10/paper/viewPaper/2196

[Barwise et al. 1980] **BARWISE**, J. and **PERRY**, J. "The situation underground" In: *Stanford Working Papers in Semantics, Stanford University Press*, 1980. Vol. 1.

[Barwise et al. 1983] **BARWISE**, J., and **PERRY**, J. Situations and Attitudes. MIT Press, 1983. Cambridge, Mass.

{Becha et al. 2012} **BECHA**,H. and A**MYOT**,D. "Non-Functional Properties in Service Oriented Architecture – A Consumer's Perspective" in journal of software, vol. 7, no. 3, march 2012. pp.575-581.

[Bellik 1995] **BELLIK, Y.** Interfaces Multimodales : Concepts, Modèles et Architectures. PhD Thesis, University Paris-South 11, Orsay, 1995. Available (04/02/2012) at: http://www.limsi.fr/-bellik/yacine/doku.php?id=publications

[Bellik et al. 1992] **BELLIK**, Y. and **TEIL**, D. "Définitions Terminologiques pour la communication multimodale". In: *Actes des 4emes Journées sur l'Ingénierie des Interfaces Homme-Machine, OHM'92*. November 30- December 2, 1992. Telecom Paris Publ., Paris. pp. 347-371. Available (04/02/2012) at: http://www.limsi.fr/-bellik/yacine/lib/exe/fetch.php?media=publis:1992_ihm_1.pdf

[Bellik et al. 1995] **BELLIK**,Y., **FERRARI**, S., **NEEL**, F.,**TEIL**, D., **PIERRE**, E. and **TACHOIRES**, V. "Interaction Multimodale : Concepts et Architecture", In: *Proceedings of the 4èmes Journées Internationales : Interface des Mondes Réels et Virtuels*. Montpellier, 26-30 Juin, 1995. 9 pages. Available (04/02/2012) at: http://www.limsi.fr/-bellik/yacine/lib/exe/fetch.php?media=publis:1992_ihm_1.pdf

{Bennett et al. 2004} **BENNETT**, B. and G**ALTON**, **A.P.** "A unifying semantics for time and events" In: *Artificial Intelligence*. March 2004. Vol. 153. PP. 13-48. http://dx.doi.org/10.1016/j.artint.2003.02.001

[Berners-Lee 1997] Axioms of Web Architecture: Metadata . Available at: http://www.w3.org/DesignIssues/Metadata.html

[Bernsen 1993] **BERNSEN**, O. "Foundations of multimodal representations. A taxonomy of representational modalities". In: *Interacting with Computers,* 1993, Vol. 6, No. 4, p. 347-371. http://dx.doi.org/10.1016/0953-5438(94)90008-6

[Bernsen et al. 2010] **BERNSEN**, O. and **DYBKJAER**, L. Multimodal Usability. Springer, 2010. 431 Pages.

[Bettini et al. 2009] **BETTINI**, C., **BRDICZKA**, O., **HENRICKSEN**, K., **INDULSKA**, J., **NICKLAS**, D., **RANGANATHAN**, A. and **RIBONI**,D. "A survey of context modelling and reasoning techniques"," In: *Pervasive and Mobile Computing*. April 2010. Vol. 6, No.2. p.p.161-180. http://dx.doi.org/10.1016/j.pmcj.2009.06.002

[Bohus et al. 2010] **BOHUS**, D. and **HORVITZ**, E. "On the Challenges and Opportunities of Physically Situated Dialog" In: *AAAI Fall Symposium on Dialog with Robots*, Arlington, VA, 2010. pp. 1-6. Available (04/02/2012) at: http://research.microsoft.com/en-us/um/people/horvitz/phys_situated_dialog.pdf

[Bolchini et al. 2007] **BOLCHINI**, C., **CURINO**, C.A., **QUINTARELLI**, E., **SCHREIBER,** F.A. and **TANCA**, L. "A data-oriented survey of context models" In: SIGMOD December 2007. Vol. 36. p.p.19-26. http://doi.acm.org/10.1145/1361348.1361353

[Bozdag et al. 2006] **BOZDAG**,E., **MESBAH**,A. and **VAN DEURSEN**, A. "A Comparison of Push and Pull Techniques for AJAX" Report TUD-SERG-2007-016a. TUDelft.

[Bolt 1980] **BOLT,** Richard A. " 'Put-that-there': Voice and gesture at the graphics interface", In: *ACM SIGGRAPH Computer Graphics*, July 1980. Vol.14 No.3, pp.262-270. http://dx.doi.org/10.1145/965105.807503

[Brdiczka et al. 2010] **BRDICZKA**, O., **CROWLEY**, J.L. and **REIGNIER, P.** "Learning Situation Models in a Smart Home" In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*. Feb. 2009. Vol.39, No.1. pp.56-63, http://dx.doi.org/10.1109/TSMCB.2008.923526

[Bredin et al. 2007] **BREDIN**, H. and **CHOLLET**, G. "Audiovisual speech synchrony measure: application to biometrics" In: *EURASIP Journal on Advances in Signal Processing*, 2007. Vol. 2007. pp. 1-11. http://dx.doi.org/10.1155/2007/70186

[Brennan 1990] **BRENNAN**, Susan E. "Conversation as direct manipulation: an iconoclastic view." In: *The art of human-computer interface design*, Edited by B. Laurel and S. J. Mountford. 1990. Addison-Wesley Pub. Co. pp.393-404.

[Bringsjord et al. 2007] **BRINGSJORD**, S. and **ARKOUDAS**, K. "The Philosophical Foundations of Artificial Intelligence" In: FRANKISH & RAMSEY, Eds. The Cambridge Handbook of Articial Intelligence, Cambridge University Press, 2012. Cambridge, UK.

[Brooks et al. 1998] **BROOKS**, R., **BREZEAL**, C., **IRIE**, R., **KEMP**, C., **MAJANOVIC,** M., **SCASSELLTATI**, B. and **WILLIAMSON**, M. "Alternative Essences of Intelligence". In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, 1998. Menlo Park, CA. pp. 961–967.

[Buchholz et al. 2004] **BUCHHOLZ** S., **HAMANN**, T. and **HUBSCH** G. "Comprehensive structured context profiles (CSCP): design and experience." In: *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, March 2004. pp. 43-47.

[Bucur et al. 2006] **BUCUR**, O., **BEAUNE**, P., **BOISSIER**, O., **ROTH-BERGHOFER**, T. and **SCHULZ**, S. "Steps Towards Making Contextualized Decisions: How to Do What You Can, with What You Have, Where You Are" In: Lecture Notes in Computer Science, 2006.Springer Berlin / Heidelberg. Vol. 3946. p.p.62-85. http://dx.doi.org/10.1007/11740674_5

[Bui 2006] **BUI,** T.H. Multimodal Dialogue Management - State of the Art. Technical Report TR-CTIT-06-01, Centre for Telematics and Information Technology University of Twente, Enschede. ISSN 1381-3625. Available (04/02/2012) at: http://www.ub.utwente.nl/webdocs/ctit/1/0000015d.pdf

[Bunt et al. 2005] **BUNT**, H.., **KIPP**, M., **MAYBURY**, M and **WAHLSTER**, W. "Fusion and Coordination for Multimodal Interaction." In: *Multimodal Intelligent Information Presentation*. STOCK, O., ZANCANARO, M. (Eds.). Series Text, Speech and Language Technology, Kluwer Academic , 2005. Vol. 27, pp. 325-340.

[Buxton 1986] **BUXTON**, W. "There's More to Interaction than Meets the Eye: Some Issues in Manual Input" In: *User Centered System Design: New Perspectives on Human-Computer Interaction*. Norman, D. A. and Draper, S. W. (Ed.). Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1986. p.p. 319-337.

[Card et al. 1991] **CARD,S. K., MACKINLAY,J. D. et ROBERTSON**, G. G. "A morphological analysis of the design space of input devices." ACM Trasactions on Information Systems, 9(2) :99–122, April 1991.

[Cariani 1992] **CARIANI**, P. "Some epistemological implications of devices which construct their own sensors and effectors" In: *Varela F, Bourgine P eds. Towards a practice of autonomous systems. MIT Press, 1991.* Cambridge, MA. pp. 484-493.

[Carmagnola 2011]	**CARMAGNOLA,** F., **CENA**, F. and **GENA**, C. "User model interoperability: a survey" In: *User Modeling and User-Adapted Interaction*, 2011. Vol.21. No.3. pp.285-331. http://dx.doi.org/10.1007/s11257-011-9097-5
[Caschera et al. 2010]	**CASCHERA**, M.C., **FERRI**, F. and **GRIFONI**, P. "An Approach for Managing Ambiguities in Multimodal Interaction" In: *Proceedings of the 2007 OTM confederated international conference on On the move to meaningful internet systems*, SpringerLink. Vol. 1.
[Cassell 2001]	**CASSELL**, J. "Embodied Conversational Agents. Representation and Intelligence in User Interfaces" In: *AI Magazin*e, 2001. Vol. 22. No.4. Available (04/02/2012) at: http://www.aaai.org/ojs/index.php/aimagazine/article/view/1593
[Cassell 2009]	**CASSELL**, J."Social Practice: Becoming Enculturated in Human-Computer Interaction" In: *Lecture Notes in Computer Science, Universal Access in Human-Computer Interaction. Applications and Services*.Springer Berlin / Heidelberg, 2009. Vol.5616. pp. 303-313. ISBN 978-3-642-02712-3 http://dx.doi.org/10.1007/978-3-642-02713-0_32
[Chai et al. 2004]	**CHAI**, J. and **PRASOV**, Z. "Cognitive Principles in Robust Multimodal Interpretation" In: *Journal of Artificial Intelligence Research*, 2006. Vol. 27. pp. 55-83. http://dx.doi.org/10.1613/jair.1936
[Chen et al. 2004a]	**CHEN**, H., **FININ**, T. and **JOSHI**, A. "A Context Broker for Building Smart Meeting Rooms" In: *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004* AAAI Spring Symposium (2004) Available (04/02/2012) at: http://ebiquity.umbc.edu/get/a/publication/78.pdf
[Chen et al. 2004b]	**CHEN**,H., **PERICH**, F., **FININ**, T. and **JOSHI**, A. "SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications" In: *Ontologies for Agents: Theory and Experiences*. Birkhauser, Basel, 2005. . ISBN: 3-7643-7237-0. Available (04/02/2012) at:http://ebiquity.umbc.edu/v2.1/_file_directory_/papers/105.pdf
[Choudhury et al. 2002]	**CHOUDHURY**, T., **REHG**, J.M., **PAVLOVIC**, V. and **PENTLAND**, A. "Boosting and structure learning in dynamic bayesian networks for audio-visual speaker detection". In: *The 16th International Conference on Pattern Recognition ICPR'02,* Quebec, 2002. Vol. 3, pp. 789–794. Available (04/02/2012) at: http://dl.acm.org/citation.cfm?id=839291.842788
[Christopoulou et al. 2005]	**CHRISTOPOULOU**,E., **GOUMOPOULOS**,C., **ZAHARAKIS**, I. and **KAMEAS**, A. "An Ontology-based Conceptual Model for Composing Context-Aware Applications" In: *6th International Conference on ubiquitous computing, Workshop on advanced context modeling, reasoning and management*. Nottingham, England. 2004. Available (04/02/2012) at : http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.6050&rep=rep1&type=pdf
[Cohen et al. 1979]	**COHEN**, P. R. and **PERRAULT**, C. R. " Elements of a plan based theory of speech acts". In: *Cognitive Science*, 1979. Vol.3 No.3. pp.177–212.
[Cohen 1997]	**COHEN,** P. "Dialogue Modeling" In: *Survey the State of the art in Human Language Technology*, 1997.
[Collier 2004]	**COLLIER**, G. Emotional Expression. Lawrence Erlbaum & Associates, 1985. 264 Pages.
[Concolato et al. 2009]	**CONCOLATO** C., **LE FEUVRE** J. and **DUFOURD** J. C. "Declarative Interfaces for Dynamic Widgets Communications" In: *Proceedings of the 9th ACM Symposium on Document Engineering*, Munich, Germany, pp. 241–244. http://dx.doi.org/10.1145/1600193.1600245
[Coutaz et al. 1991]	**COUTAZ**, J. and **BALBO**, S. "Applications: a dimension space for user interface management systems" In: P*roceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology.* April 27-May 02, 1991, New Orleans, USA. pp.27-32. http://dx.doi.org/10.1145/108844.108849
[Coutaz et al. 1995]	**COUTAZ**, J., **NIGAY**, L., **SALBER**, D., **BLANDFORD,** A., **MAY**, J. and **YOUNG**, R. "Four easy pieces for assessing the usability of multimodal interaction: the CARE properties" In: *Proceedings of INTERACT'95* Lillehammer, June 1995
[Cristani et al. 2011]	**CRISTANI**, M., **PAGGETTI**, G., **VINCIARELLI**, A., **BAZZANI**, L., **MENEGAZ**, G. and **MURINO**, V. "Towards Computational Proxemics: Inferring Social Relations from Interpersonal Distances" In: Proceedings of IEEE International Conference on Social Computing, 2011. pp. 290-297. http://dx.doi.org/10.1109/PASSAT/SocialCom.2011.32
[Crofts et al. 2009]	**CROFTS**, N., **DOERR**, M., **GILL**, T., **STEAD**, S. and **STIFF**, M. (Ed.) "Definition of the CIDOC Conceptual Reference Model" December 2011. Available (04/02/2012) at: http://www.cidoc-crm.org/official_release_cidoc.html
[Crowley et al. 2002]	**CROWLEY**,J., **COUTAZ**, J., **REY**,G. and **REIGNIER**, P. "Perceptual components for context aware computing" In: *Proceedings of the 4th international conference on Ubiquitous Computing UbiComp '02*. Gaetano Borriello and Lars Erik Holmquist (Eds). Springer-Verlag, London, UK. p.p. 117-134. Available (04/02/2012) at : http://www.springerlink.com/index/CYG6J0TX829CJC8E.pdf

[Crowley et al. 2006] **CROWLEY** J., **BRDICZKA**, O. and **REIGNIER**, P. "Learning Situation Models for Understanding Activity" In: *International Conference on Development and Learning, ICDL 06.* June 2006. Available (04/02/2012) at : http://www.mentaldev.org/icdl06/paper/104_Crowley_paper.pdf

[D'Ulizia 2009] **D'ULIZIA**, A. "Exploring Multimodal fusion Strategies" In : Multimodal human computer interaction and pervasive services. **GRIFONI**, P. (Editor). Hershey, NY, 2009. 539 pages.

[Dasarathy 1997] **DASARATHY**, B. "Sensor Fusion Potential Exploitation-Innovative Architectures and Illustrative Applications" In: *Proceedings of the IEEE*, 1997. Vol. 85. No.1. pp 24–38. http://dx.doi.org/10.1109/5.554206

[Dees et al. 2007] **DEES**, W. and **SHRUBSOLE**, P. "Web4CE: accessing web- based applications on consumer devices". In: *Proceedings of the 16th international Conference on World Wide Web* (Banff, Alberta, Canada, May 08 - 12, 2007). WWW '07. ACM, New York, NY, 1303-1304. http://dx.doi.org/10.1145/1242572.1242820

[Descartes 1637] **DESCARTES**, R. "La Géométrie" In: Discours de la méthode, 1637. Available (04/02/2012) at : http://ia700200.us.archive.org/19/items/geometryofrene00desc/geometryofrene00desc.pdf

[Dey 2001] **DEY, A. K.** *"Understanding and Using Context"* In: Personal and Ubiquitous Computing, Fev. 2001. Springer, London. Vol. 5. No. 1. pp.4-7. http://dx.doi.org/10.1007/s007790170019

[Dibowski et al. 2011] **DIBOWSKI**, H. and **KABITZSCH,** K. "Ontology-Based Device Descriptions and Device Repository for Building Automation Devices". In: *EURASIP Journal of Embeeded Systems,* 2011. Vol.2011. pp 1–17. http://dx.doi.org/10.1155/2011/623461

[Ding et al. 2007] **DING**, X. and **IIJIMA**, J. "A Framework for Modeling Situation Dependent Service Process" In: *International Conference on Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007.* 21-25 Sept. Shanghai. p.p. 3829-3832. http://dx.doi.org/10.1109/WICOM.2007.947

[Dobson et al. 2006] **DOBSON**, S. and **YE**, J. "Using fibrations for situation identification" In: *Proceedings of Pervasive 2006 workshops*. Dublin, Ireland. Available (04/02/2012) at : http://www.smartlab.cis.strath.ac.uk/CTSB/Dobson.pdf

[Dreyfus 1992] **DREYFUS**, H. L. What Computers still can't do. A Critique of Artificial Reason. MIT Press, 1992. Cambridge, MA.

[Dubois et al. 1991] **DUBOIS**, D. and **KONING**, J-L. "DEBORA: A Decision Engine Based On Rational Aggregation" In: *Proceedings of the Second Annual Conference on AI, Simulation and Planning in High Autonomy Systems,* 1-2 Apr 1991. pp.68-77. http://dx.doi.org/10.1109/AIHAS.1991.138450

[Duce 2000] **DUCE**, D. and **DUKE**, D. "Syndetic Modelling: Computer Science meets Cognitive Psychology". In: *Electronic Lecture Notes in Theoretical Computer Science*, 2000. Vol. 43. Invited lecture at Formal Methods Elsewhere, a satellite workshop of FORTE-PSTV-2000. doi:10.1016/S1571-0661(04)80894-6

[Dumas et al. 2009] **DUMAS**, B., **LALANNE**, D. and **OVIATT**, S. "Multimodal Interfaces: A Survey of Principles, Models and Frameworks" In: *Human Machine Interaction: Research Results of the MMI Program*, Springer-Verlag, Berlin, Heidelberg, 2009 http://dx.doi.org/10.1007/978-3-642-00437-7_1

[Eagle et al. 2006] **EAGLE**, N. and **PENTLAND**, A. "Reality mining: sensing complex social signals" In: *Journal of Personal and Ubiquitous Computing*. March 2006. Vol. 10. No. 4. pp. 255–268. http://dx.doi.org/10.1007/s00779-005-0046-3

[Egenhofer 1989] **EGENHOFER**, M. "A formal definition of Binary Topological Relationships" In: *Third International Conference on Foundations of Data Organization and Algorithms FODO*. LITWIN, W. & SCHEK, H. (Eds.) Paris, France, 1989. Available (04/02/2012) at: http://www.spatial.maine.edu/~max/fodo.pdf

[Ejigu 2007] **EJIGU,** D. Services Pervasifs Contextualisés: Modélisation et Mise en Œuvre. PhD Thesis, INSA de Lyon, 2007.

[Esposito et al. 2009] **ESPOSITO**, F., **BASILE**. T. M. A., **DI MAURO**, N. and **FERILLI**, S. "Machine Learning Enhancing Adaptivity of Multimodal Mobile Systems" In: *Multimodal Human Computer Interaction and Pervasive Services* (GRIFONI, P. Ed). Information Science Reference, 2009. pp.121-138.

[EVANS-YAML 2001] YAML Ain't Markup Language http://www.yaml.org/

[Fitzgerald et al. 2003] **FITZGERALD**, W. and **FIRBY** R. J. "Multimodal Event Parsing for Intelligent User Interfaces." In : *Proceedings of the 8th international conference on Intelligent user interfaces IUI '03.* Miami, Florida, 2003. pp.53-60.. http://dx.doi.org/10.1145/604045.604058

[Foley 1984] **FOLEY** J.D., **WALLACE**,V.L. and **CHAN, P**. "The Human Factors of computer Graphics interaction techniques" In: *IEEE computer Graphics and Applications*, 1984. Vol. 4. No.11, pp. 13-48.

[Foster 2002] **FOSTER,** M.E. "State of the art review: Multimodal fission." COMIC project Deliverable 6.1, September 2002. Project Number IST-2001-32311. University of Edinburgh.

[Fowler 1998] **FOWLER**, M. Analysis Patterns: Reusable Object Models. Addison-Wesley, 1997.

[Frank 1992] **FRANK**, A.U. "Qualitative Spatial Reasoning about Distances and Directions in Geographic Space" In: *Journal of Visual Languages and Computing,* 1992. Vol. 3. p.p. 343-371. University of Maine, Orono, Maine, U.S. http://dx.doi.org/10.1016/1045-926X(92)90007-9

[Frank 1998] **FRANK**, A.U. "Formal Models for Cognition - Taxonomy of Spatial Location Description and Frames of Reference" In: *Spatial Cognition* 1998. FREKSA, C. HABEL,C. and WENDER, K. F. (Ed.) p.p. 293-312. http://dx.doi.org/10.1007/3-540-69342-4_14

[Frank 2001] **FRANK**, A.U. "Tiers of ontology and consistency constraints in geographical information systems" In: *International Journal of Geographical Information Science*, 2001. Vol. 15. No. 7. p.p. 9-77. http://dx.doi.org/10.1080/13658810110061144

[Frank 2003] **FRANK**, A.U. "Ontology for Spatio-temporal Databases" In: *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, 2003. ISBN 978-3-540-40552-8. p.p. 9-77. http://dx.doi.org/10.1007/978-3-540-45081-8_2

[Freksa 1992] **FREKSA**.C. "Temporal reasoning based on semi-intervals" In: *Artificial Intelligence*, 1992. Vol.54. No.1. p.p. 199–227. http://dx.doi.org/10.1016/0004-3702(92)90090-K

[Gaitanis et al. 2007] **GAITANIS**, K., **VYBORNOVA**, O., **GEMO**, M. and **MACQ**, B. « Multimodal High Level Fusion Of Input Commands As A Semantic Goal-Oriented Cooperative Process » In : *Proceedings of the 12th International Conference on Speech and Computer*. October 15-18, 2007. Moscow, Russia.

[Gangemi et al. 2002] **GANGEMI**, A., **GUARINO**, N., **MASOLO**, C., **OLTAMARI**, A. and **SCHNEIDER**, L. "Sweetening Ontologies with DOLCE". In: *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02),* Oct. 14, 2002. Sigüenza, Spain. Lecture Notes in Computer Science. Vol. 2473 page 166. http://dx.doi.org/10.1007/3-540-45810-7_18

[Garg et al. 2003] **GARG,** A., **PAVLAVIC**.,V. and **REHG**, J. M. "Audio-visual speaker detection using dynamic bayesian networks" In: *Proceedings of the Fourth Internarional Conference on Automatic Face and Gesture Recognition*, March 28-30,2000. Grenoble, France. pp. 384-390. http://dx.doi.org/10.1109/AFGR.2000.840663

[Garlan et al. 2002] **GARLAN**, D., **SIEWIOREK**,D., **SMAILAGIC**,A. and **STEENKISTE**, P. "Project Aura: Toward Distraction-Free Pervasive Computing", In: *IEEE Pervasive Computing*, vol. 1, no. 2, pp. 22-31, Apr.-June 2002. http://doi.ieeecomputersociety.org/10.1109/MPRV.2002.1012334

[Gatica-Perez 2009] **GATICA-PEREZ**, D. "Automatic nonverbal analysis of social interaction in small groups: a review" In: *Image and Vision Computing*, 2009. Vol. 27. No. 12. pp. 1775–1787. http://dx.doi.org/10.1016/j.imavis.2009.01.004

[Gauss 1827] **GAUß**, C.F. Disquisitiones generales circa superficies curvas, Göttingen, 1828. Available (04/02/2012) at: http://gdz.sub.uni-goettingen.de/dms/load/img/?PPN=PPN236005081&IDDOC=139369

[Gellersen et al. 2002] **GELLERSEN**, H.W., **SCHMIDT**, A. and **BEIGL,** M. "Multi-sensor context-awareness in mobile devices and smart artifacts" In: *Mobile Networks and Applications*, October 2002. Vol. 7. No. 5. p.p.341-351. http://dx.doi.org/10.1023/A:1016587515822

[Gerevini et al. 2002] **GEREVINI**, A. and **NEBEL**, B. "Qualitative Spatio-Temporal Reasoning with RCC-8 and Allen's Interval Calculus: Computational Complexity" In: *Proceedings of the 15th European Conference on Artificial Intelligence ECAI02* (2002) Available (04/02/2012) at: ftp://ftp.informatik.uni-freiburg.de/documents/papers/ki/gerevini-nebel-ecai02.pdf

[Getis et al. 1978] **GETIS**, A. and **BOOTS**, B. Models of Spatial Processes - An approach to the study of point, line and area patterns. Cambridge Geographical Studies. Cambridge University Press, Cambridge, 1978. ISBN-10: 0521103541

[Ghazarian et al. 2010] **GHAZARIAN**, A. and **NOORHOSSEINI**, S.M. "Automatic detection of users' skill levels using high-frequency user interface events." In: *User Modeling and User-Adapted Interaction*. Kluwer Academic Publishers, 2010. Hingham, MA, USA. Vol.20. No.2. pp.109-146. http://dx.doi.org/10.1007/s11257-010-9073-5

[GOOGLE-AINTENT 2012] **GOOGLE.** Android Intent Class Documentation Available at URL: http://developer.android.com/reference/android/content/Intent.html Visited at Sept 2012.

[Greene 2008] **GREENE,** K. "Special Report: 10 Emerging Technologies." In: *MIT Technology Review,* February 2008. Available (04/02/2012) at: http://www.technologyreview.com/tr10/

[Gripay et al. 2006] **GRIPAY**, Y., **PIERSON** J., **PIGEOT** C.E. "Une architecture pervasive sécurisée : PerSE", In: *UbiMob'06*, ACM ed., 2006, Paris. pp. 147-150

[Guarino 2004] **GUARINO,** N., "Toward a Formal Evaluation of Ontology Quality". IEEE intelligent Systems, 2004. 19(4): 78-79. Guarino, N., and Musen, M. Focusing on Content

[Habermas 1981] **HABERMAS**, J. Theory of Communicative Action. Thomas McCarthy (Trans.) Beacon Press, Boston, 1984

[Halpin 2009] **HALPIN**, T. "Object-Role modeling." In: *Encyclopedia of Database Systems*, 2009. Springer, 2009.

[Hall et al. 1997] **HALL**, D.L and **LLINAS** J. "An Introduction to Multisensor Data Fusion", In: *Proceedings of the IEEE*, 1997. Vol. 85. No 1. pp. 6-23 http://dx.doi.org/10.1109/5.554205

[Hallot 2006] **HALLOT**, P. Relations spatio−temporelles dans un espace−temps primitif: Un essai de simplification de l'analyse spatio−temporelle. PhD Thesis, Université de Liège, Sciences géographiques, 2006. Available (04/02/2012) at: http://orbi.ulg.ac.be/handle/2268/690

[Haspelmath 2006] **HASPELMATH**, M. From Space to Time Temporal Adverbials in the World's Languages. Lincom Studies in Theoretical Linguistics, Lincom Europa, Munich & Newcastle. Vol. 3. 181 pp.Available at: http://email.eva.mpg.de/-haspelmt/SpaceTime.pdf

[Haugeland 1985] **HAUGELAND**, J. Artificial Intelligence: The Very Idea, MIT Press, 1985. Cambridge, MA. ISBN: 0-262-08153-9

[Hayes 1978] **HAYES**, P. J. The naive physics manifesto. In: *Expert Systems in the Micro-Electronic Age*. MICHIE, D. (Ed.) Edinburgh University Press, 1978. ISBN-10: 0852244932.

[Heckmann et al. 2005] **HECKMANN**, D., **SCHWARTZ**, T., **BRANDHERM**, B. and **KRÖNER**, A. "Decentralized user modeling with UserML and GUMO" In: *Proceedings of the Workshop on Decentralized, Agent Based and Social Approaches to User Modeling, DASUM-05, at UM2005.* July,2005. Edinburgh, Scotland, pp. 61–66 Available (04/02/2012) at: http://www.cs.aau.dk/-dolog/dasum/DASUM05.Heckmann.pdf

[Helbing et al. 1998] **HELBING**, D. and **MOLNÁR**, P. "Social force model for pedestrian dynamics" Physical Review E,1995. Vol. 51. No. 5. pp. 4282–4287. http://dx.doi.org/10.1103/PhysRevE.51.4282

[Henricksen et al. 2006] **HENRICKSEN**, K. and **INDULSKA**, J. "Developing context-aware pervasive computing applications: Models and approach" In: *Pervasive and Mobile Computing*, February 2006. Vol. 2. No.1. p.p. 37–64 http://dx.doi.org/10.1016/j.pmcj.2005.07.003

[Hernandez 2006] **HERNANDEZ**, Nathalie. Ontologies de Domaine pour la Modélisation du Contexte en Recherche d'Information. PhD Thesis, Université Paul Sabatier, december 2005. http://www.irit.fr/-Nathalie.Hernandez/nHernandez.pdf

[Hofer et al. 2009] **HOFER**, B. and **FRANK**, A.U. "Towards a Method to Generally Describe Physical Spatial Processes" In: *Proceedings of the SDH 2008 conference: Headway in Spatial Data Handling*, 2009. Montpellier France. p.p 217-232. http://dx.doi.org/10.1007/978-3-540-68566-1_13

[Hung et al. 2007] **HUNG**, H., **JAYAGOPI**, D., **YEO**, C., **FRIEDLAND**, G., **BA**, S., **ODOBEZ**, J.M., **RAMCHANDRAN**, K., **MIRGHAFORI**, N. and **GATICA-PEREZ**, D. "Using audio and video features to classify the most dominant person in a group meeting" In: *Proceedings of the ACM International Conference on Multimedia*, 2007, pp. 835–838.

[Hutchins et al. 1985] **HUTCHINS**, E. L., **HOLLAN**, J.D. and **NORMAN**, D. A. "Direct Manipulation Interfaces". In: Human-Computer Interaction, Volume 1, pp. 311-338. Lawrence Erlbaum Associates, 1985.

[Jacob et al. 1992] **JACOB**, R. J. K. and **SIBERT**, L. E. "The perceptual structure of multidimensional input device selection" In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI92*, 1992. ACM, New York, NY, US. http://doi.acm.org/10.1145/142750.142792

[Jaimes et al. 2007] **JAIMES**, A. and **SEBE**, N. "Multimodal human-computer interaction: A survey". In: *Computer Vision and Image Understanding.* Elsevier Science Inc. New York, NY, USA . 2007 http://dx.doi.org/10.1016/j.cviu.2006.10.019

[Jakkula et al. 2007] **JAKKULA**, V. and **COOK**, D. "Using temporal relations in smart environment data for activity prediction" In: *Proceedings of the International Conference on Machine Learning-ICML. Workshop on the Induction of Process Models-IPM.* Corvalis, June 2007. Available (04/02/2012) at: http://eecs.wsu.edu/-vjakkula/ICML.pdf

[Johnson-Laird 1983] **JOHNSON-LAIRD**, P.N. Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness. Cambridge University Press, 1983.

[Kaenampornpan 2009] **KAENAMPORNPAN**, M. A Context Model, Design Tool and Architecture for Context-Aware Systems Design. PhD Thesis. University of Bath Department of Computer Science, 2009. Available (04/02/2012) at: http://opus.bath.ac.uk/17197/

[Kagal et al. 2007] **KAGAL** L., **KOROLEV** V., **AVANCHA** S., **JOSHI**,A, **FININ**,T and **YESHA**, Y. "Centaurus: an infrastructure for service management." In: *Ubiquitous Computing Environments. Wireless Networks,* November 2002. Vol. 8 No.6. pp.619-635.

[Kay 1994] **KAY**, J. "Lies, Damned Lies and Stereotypes: Pragmatic Approximations of Users." In: *Proceedings of Fourth International Conference on User Modeling*, 1994. pp. 175-184.

[Khriyenko et al. 2005] **KHRIYENKO** O. and **TERZIYAN** V. "Context Description Framework for the Semantic Web." In: *Context Representation and Reasoning Workshop*, Paris, France, July 2005.

[Kinlan 2010] **KINLAN**, P. Web Intents Specification. Available (04/02/2012) at: https://github.com/PaulKinlan/WebIntents and http://webintents.org/

[Kobsa 1989] **KOBSA**, A. and **WAHLSTER**, W. User models in dialog systems. Springer-Verlag, 1989.

[Kobsa 1993] **KOBSA**, A. "User modeling: Recent Work, Prospects and Hazards" In: Adaptive User Interfaces: Principles and Practice, North-Holland, Amsterdam, 1993. New York, NY, USA. ISBN 0444815457.

[Kontchakov et al. 2007] **KONTCHAKOV**, A., **KURUCZ**, A., **WOLTER**, F. and **ZAKHARYASCHEV**, M. "Spatial logic + temporal logic = ?" In: Handbook of Spatial Logics, 2007. AIELLO, M., PRATT–HARTMANN, I. & VAN BENTHEM, J. (Ed.) Springer Netherlands,Kluwer, Dordrecht. p.p.497-564. Isbn: 978-1-4020-5587-4 http://dx.doi.org/10.1007/978-1-4020-5587-4_9

[Knapp et al. 2009] **KNAPP**, M. L. and **HALL**, J. A. Nonverbal Communication in Human Interaction. Thomson Wadsworth, 2009. (7th. edition)

[Kuflik et al. 2009] **KUFLIK**, T., **KAY**, J. and **KUMMERFELD**, B. "Challenges and Solutions of Ubiquitous User Modeling." In: *German-Israeli Minerva School for Ubiquitous Display Environments: Intelligent Group Interaction, Foundations and Implementation of Pervasive Multimodal Interfaces*, August-September 2009. Available (04/02/2012) at: http://www.cri.haifa.ac.il/crievents/2009/details/160-german-israeli-minerva-school-for-ubiquitous-display-environments-intelligent-group-interaction-foun

[Lagoze et al. 2001] **LAGOZE**, C. and **HUNTER**, J. "The ABC ontology and model" In: *Proceedings of the International Conference on Dublin Core and Metadata Applications - DMCI 2001*, National Institute of Informatics, Tokyo, Japan, 2001. p.p. 160-176 Available (04/02/2012) at: http://www.nii.ac.jp/dc2001/proceedings/abst-26.html

[Lambert 2004] **LAMBERT**, D. Body Language. HarperCollins, 2004.

[Larsson 2005] **LARSSON**, S. "Dialogue Systems: Simulations or Interfaces?" In: *Gardent & Gaiffe Ed., Proceedings of the ninth workshop on the semantics and pragmatics of dialogue, DIALOR*. June 9-11, 2005. Nancy, France. Available (04/02/2012) at: http://dialor05.loria.fr/Papers/06-larsson_final.pdf

[Liu et al. 2006] **LIU**, H., **MAES**, P., and **DAVENPORT,** G. "Unraveling the taste fabric of social networks". In: *International Journal on Semantic Web and Information Systems,* 2006. Vol.2. No.1. pp. 42–71.

[Loke 2011] **LOKE**, S.W. "On representing situations for context-aware pervasive computing: six ways to tell if you are in a meeting" In: *4° Annual IEEE International Conference on Pervasive Computing and Communications Workshops-PerCom Workshops Pisa, 2006.* p.p.5-39. http://dx.doi.org/10.1109/PERCOMW.2006.102

[Longacre 1983] **LONGACRE**, R. E. The grammar of discourse. Plenum, New York, 1983. ISBN-10: 0306452359

[Lynch 2011] **LYNCH,** L. "Inside the Identity Management Game" In: *Internet Computing, IEEE*. Sept.-Oct. 2011. Vol. 15. No. 5. http://dx.doi.org/10.1109/MIC.2011.119

[Latta et al. 1994] **LATTA,** J. and **OBERG**, D. "A Conceptual Virtual Reality Model" In: *IEEE Computer Graphics and Applications, January 1994*. Vol. 14. No. 1. pp.23-29. doi:10.1109/38.250915

[Ligozat et al. 2006] **LIGOZAT**, G. and **CONDOTTA**,J.-F. "On the relevance of conceptual spaces for spatial and temporal reasoning" In: *Spatial Cognition and Computation*,2006. Vol. 5. No.1. p.p. 1-27. http://dx.doi.org/10.1207/s15427633scc0501_1

[Mackinlay 1990] **MACKINLAY**, J., **CARD**,S. and **ROBERTSON**, G. "A Semantic Analysis of the Design Space of Input Devices" In: *Human Computer Interaction*, 1990. Lawrence Erlbaum, Vol. 5. No. 2-3. pp. 145-190. http://dx.doi.org/10.1207/s15327051hci0502&3_2

[McCarthy et al. 1998] **MCCARTHY**, J and **SASA**. B. "Formalizing context (expanded notes) on Computing Natural Language." In: *Computing natural language, CSLI Lecture Notes*. Stanford, Californie, 1998, Vol. 81. pp. 13-50.

[McCowan et al. 2005] **MCCOWAN**, I., **GATICA-PEREZ**, D., **BENGIO**, S., **LATHOUD**, G., **BARNARD**, M. and **ZHANG**, D. "Automatic Analysis of Multimodal Group Actions in Meetings" In: *Ieee Transactions on Pattern Analysis and Machine Intelligence*, March 2005. Vol. 27. No. 3. http://dx.doi.org/10.1109/TPAMI.2005.49

[Maes et al. 2003] **MAES,** S. H. and **SARASWAT,** V. Multimodal Interaction Requirements. World Wide Web Consortium, W3C Note, 8 January 2003. Available (04/02/2012) at: http://www.w3.org/TR/2003/NOTE-mmi-reqs-200301087

[Mattioli et al. 2007] **MATTIOLI**, J., **MUSEUX**, N., **HEMAISSIA**, N. and **LAUDY**, C. "A Crisis Response Situation Model" In: 10th International Conference on Information Fusion, 2007. p.p. 1-7. http://dx.doi.org/10.1109/ICIF.2007.4408022

[Martin 1995] **MARTIN**, J-C. Coopérations entre Modalités et Liage par Synchronie dans les Interfaces Multimodales, PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, Mars 1995.

[Martin et al. 2001] **MARTIN**, J.C., **GRIMARD**, S., and **ALEXANDRI**, K. "On the annotation of the multimodal behavior and computation of cooperation between modalities." In: *Proceedings of the Workshop on Representing, Annotating, and Evaluating Non-Verbal and Verbal Communicative Acts to Achieve Contextual Embodied Agents*, Montreal, Canada. pp. 1-7.

[Masuoka et al. 2003] **MASUOKA** R., **PARSIA** B., and **LABROU**, Y. "Task computing - the semantic web meets pervasive computing". In: *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia - MUM04*. pages 866–881, 2003. http://doi.acm.org/10.1145/1052380.1052416

[Michalowski et al. 2006]	**MICHALOWSKI**, M.P., **SABANOVIC**, S. and **SIMMONS**, R. "A spatial model of engagement for a social robot" In: *9th IEEE International Workshop on Advanced Motion Control*. Istanbul, 2006. p.p.762-767. http://dx.doi.org/10.1109/AMC.2006.1631755
[Millard et al. 2004]	**MILLARD**,I., **DE ROURE**, D. and **SHADBOLT**,N. "The use of ontologies in contextually aware environments" In: *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management*, University of Southampton, Nottingham, England, 2004. Available (04/02/2012) at: http://eprints.ecs.soton.ac.uk/9973/
[Merckel et al. 2009]	**MERCKEL**,L. and **NISHIDA**,T. "Enabling Situated Knowledge Management for Complex Instruments by Real-time Reconstruction of Surface Coordinate System on a Mobile Device" In: *AI & Society*, Vol. 24, No. 1, 2009, pp. 85-95. http://dx.doi.org/10.1007/s00146-009-0200-y
[Möerchen 2006a]	**MÖERCHEN**, F. "A better tool than Allen's relations for expressing temporal knowledge in interval data" In: *TDM Workshop, ACM SIGKDD, 2006*. p.p. 25–34. Available (04/02/2012) at: http://www.mybytes.de/papers/moerchen06tdm.pdf
[Möerchen 2006b]	**MÖERCHEN**, F. Time Series Knowledge Mining. Phd Thesis, Philipps-University Marburg, Germany, 2006. Görich & Weiershäuser. ISBN 3-89703-670-3 Available (04/02/2012) at: http://www.mybytes.de/papers/moerchen06tskm.pdf
[Möerchen 2007]	**MÖERCHEN**, F. "Unsupervised pattern mining from symbolic temporal data" In: *SIGKDD ACM Explorations Newsletter - Special issue on data mining for health informatics*, June 2007 Vol. 9. No.1 p.p. 41-55. http://doi.acm.org/10.1145/1294301.1294302
[Möerchen et al. 2010]	**MÖERCHEN**, F. and **FRADKIN**, D. "Robust mining of time intervals with semi-interval partial order patterns" In: *Proceedings of SDM'2010*. p.p.315-326. Available (04/02/2012) at: http://www.siam.org/proceedings/datamining/2010/dm10_028_moerchenf.pdf
[Mohammad et al. 2009]	**MOHAMMAD**,Y., **NISHIDA**,T. and **OKADA**, S. "Unsupervised Simultaneous Learning of Gestures, Actions and their Associations for Human-Robot Interaction" In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*. St. Louis, USA, 10-15 Oct. 2009. pp. 2537-2544. http://dx.doi.org/10.1109/IROS.2009.5353987
[Musse et al. 1997]	**MUSSE, R.D. and THALMANN, D.** «A model of human crowd behavior: Group inter-relationship and collision detection analysis» In: *Proc. CAS'97, Springer Verlag, Wien* Available (15/02/2012) at: http://archiveweb.epfl.ch/vrlab.epfl.ch/Publications/publications_index.html
[Nakauchi et al. 2000]	**NAKAUCHI**, Y. and **SIMMONS**, R. "A social robot that stands in line" In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, October 2000. Vol. 1, p.p. 357-364. http://dx.doi.org/10.1109/IROS.2000.894631
[Nefian et al. 2002]	**NEFIAN**, A.V., **LIANG**, L., **PI**, X., **LIU**, X. and **MURPHYE**, K. "Dynamic bayesian networks for audio-visual speech recognition" In : *Proceedings IEEE Int. Conf. Acoustics, Speech, Signal Processing EURA-SIP*. May 2002. Orlando, FL. Vol. 11. pp. 2013-2016. http://dx.doi.org/10.1155/S1110865702206083
[Ni et al. 2007]	**NI**, J., **MA**, X., **XU**, L. and **WANG**, J. "An image recognition method based on multiple bp neural networks fusion" In: *IEEE International Conference on Information Acquisition Speech Audio Process*, 2004. Vol. 12. No.5. pp. 520–529. http://dx.doi.org/10.1109/ICIA.2004.1373380
[Nickel et al. 2005]	**NICKEL**, K., **GEHRIG**, T., **STIEFELHAGEN**, R. and **MCDONOUGH**, J. «A joint particle filter for audio-visual speaker tracking" In: *Proceedings of the 7th International Conference on Multimodal Interfaces, ICMI'05, 2005*. Toronto, Italy. pp. 61–68. http://dx.doi.org/10.1145/1088463.1088477
[Nicklas et al. 2001]	**NICKLAS**, D. and **MITSCHANG**, B. "The nexus augmented world model: An extensible approach for mobile, spatially aware applications"," in: Proceedings of the 7th International Conference on Object-Oriented Information Systems OOIS-2001. Springer. Available (04/02/2012) at: http://www.offis.de/en/r_d_divisions/transportation/publications/publications_in_detail/info/the-nexus-augmented-world-model-an-extensible-approach-for-mobile-spatially-aware-applications.html
[Nigay 1994]	**NIGAY**, L. Conception et Modélisation Logicielles des Systèmes Interactifs : Application aux Interfaces Multimodales, PhD thesis, Université de Grenoble I, Grenoble, 1994. 315 Pages
[Nigay et al. 1993]	**NIGAY**, L. and **COUTAZ**, J. "A Desing Space for Multimodal Systems Current Procession and Data Fusion" In: *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems, 1993*. Amsterdam, April 24-29. pp. 172-178. http://dx.doi.org/10.1145/169059.169143

[Nigay et al. 1996] **NIGAY**, L. and **COUTAZ**, J. “Espaces conceptuels pour l'interaction multimédia et multimodale” In: *TSI, spécial Multimédia et Collecticiel*. AFCET & Hermes Publ., 1996. Vol. 15 No.9. pp. 1195-1225 http://iihm.imag.fr/publs/1996/TSI96-Mu.pdf

[Nishida 2010] **NISHIDA**, T. "Social Intelligence Design for Cultivating Shared Situated Intelligence" In: *Proceedings of the IEEE International Conference on Granular Computing, GrC 2010*. 14-16 Aug. 2010. San Jose, CA. pp.369-374. http://dx.doi.org/10.1109/GrC.2010.170

[Niu et al. 2010] **NIU**, W. T. and **KAY**, J. "PERSONAF: framework for personalised ontological reasoning in pervasive computing". In: *User Modeling and User-Adapted Interaction*, February 2010. Vol.20. No.1 pp.1-40. http://dx.doi.org/10.1007/s11257-009-9068-2

[Norman 1988] **NORMAN**, D. A. The Design of Everyday Things. Basic Book, New York, 1988. 272 Pages.

[Norman et al. 1986] **NORMAN**, D., **DRAPER**, S.W. 1986. User Centered System Design: New Perspectives on Human-Computer Interaction, Hillsdale, NJ: Lawrence Erlbaum Associates.

[O'Grady et al. 2011] **O'GRADY,** M.J, **DRAGONE**, M, **TYNAN**, R, **O'HARE**, G.M.P, **WAN**, J. and **MULDOON**, C. "Implicitly and Intelligently Influencing the Interactive Experience" In: Agents for Games and Simulations II: Trends in Techniques, Concepts and Design. Springer-Verlag Berlin, 2011. pp. 91. ISBN: 978-3-642-18180-1. Available (04/02/2012) at: http://irserver.ucd.ie/dspace/handle/10197/2104

[Oberle et al. 2006] **OBERLE**, D., **ANKOLEKAR**, A., **HITZLER**, P., **SINTEK**, M., **KIESEL**, M., **MOUGOUIE**, B., **VEMBU**, S., **BAUMANN**, S., **ROMANELLI,**M., **BUITELAAR**, P., **ENGEL**, R., **SONNTAG**, D., **REITHINGER,** B., **LOOS**, B., **PORZEL**, R., **ZORN**, H-P., **MICELLI**, V., **SCHMIDT**, C., **WEITEN**, M., **BURKHARDT**, F. and **ZHOU**, J. "DOLCE ergo SUMO: On foundational and domain models in the SmartWeb Integrated Ontology (SWIntO)" In: *Proceedings of 3rd International Conference on Language Resources and EvaluationConference -LREC and OntoLex2002 Workshop*. http://dx.doi.org/10.1016/j.websem.2007.06.002

{OGSA} **Foster,I., Argonne et al.** OGSA The Open Grid Services Architecture, 24 July 2006 Version 1.5. Available at URL: http://www.ogf.org/documents/GFD.80.pdf

[Ohmoto et al. 2010] **OHMOTO**, Y., **TAKAHASHI**, A., **OHASHI**, H. and **NISHIDA**, T. “Capture and Express Behavior Environment (CEBE) as WOZ system to realize effective human-agent interaction” In: *Proceedings of the International Workshop on Interacting with ECAs as Virtual Characters at AAMAS'10*. May 10-14, 2010. Toronto, Canada. pp. 63-69. http://dx.doi.org/10.1007/978-3-642-17184-0_4

[Ortony et al. 1988] **ORTONY**, A., **CLORE**, G., and **COLLINS,** A. The cognitive structure of emotions. Cambridge University Press, 1988. Cambridge, MA.

[Oviatt 2003] **OVIATT**, S. L. “Advances in Robust Multimodal Interface Design”. In: *IEEE Computer Graphics and Applications,* September 2003. Vol. 23. No. 5. http://doi.ieeecomputersociety.org/10.1109/MCG.2003.1231179

[Oviatt et al. 2001] **OVIATT**, S.L., **COHEN**, P.R., **WU**, L., **VERGO**, J., **DUNCAN**, L., **SUHM**, B., **BERS**, J., **HOLZMAN**, T., **WINOGRAD**, T., **LANDAY**, J., **LARSON**, J. and **FERRO**, D. “Designing the user interface for multimodal speech and gesture applications: State-of-the-art systems and research directions.” In: *Carroll, J. (ed.) Human-Computer Interaction in the New Millennium*. Addison-Wesley Press, Reading, 2001. Ch. 19, pp. 421–456. doi:10.1207/S15327051HCI1504_1

[Pacchierotti et al. 2006] **PACCHIEROTTI**, E. **CHRISTENSEN**,H. I. and **JENSFELT**, P. "Embodied Social Interaction for Service Robots in Hallway Environments" In: *Springer Tracts in Advanced Robotics, Field and Service Robotics*. Springer Berlin / Heidelberg, 2006. Vol. 25. pp. 293-304. http://dx.doi.org/10.1007/978-3-540-33453-8_25

[Pellan et al. 2009] **PELLAN**, B. and **CONCOLATO**, C. “Authoring of scalable multimedia documents” In: *Multimedia Tools and Applications*. July 2009. Vol. 43. No.3. pp. 225–252 http://dx.doi.org/10.1007/s11042-009-0268-x

[Pellegrini et al. 2009] **PELLEGRINI**, S., **ESS**,A., **SCHINDLER**,K. and **VanGOOL**,L. V. “You'll never walk alone: modeling social behavior for multi-target tracking” in *Proc. 12th International Conference on Computer Vision*, Kyoto, Japan, 2009, pp. 261 –268. http://dx.doi.org/10.1109/ICCV.2009.5459260

[Pentland 2007] **PENTLAND**, A. "Social signal processing" In: *Signal Processing Magazine, IEEE, July 2007*. Vol.24. No.4. pp.108-111. http://dx.doi.org/10.1109/MSP.2007.4286569

[Pinheiro 2001] **PINHEIRO DA SILVA**, P. "User Interface Declarative Models and Development Environments: A Survey" In: *Proceedings of the 7th international conference on Design, specification, and verification of interactive systems DSV-IS'00*. Available (04/02/2012) at: http://www.cs.utep.edu/paulo/papers/PinheirodaSilva_DSVIS_2000.pdf

[Pisanelli et al. 2003] **PISANELLI**, D.M., **GANGEMI**, A., **STEVE**, G. "An ontology of descriptions and situations for Lyee's hypothetical world" In: *Proceedings of Somet Workshop, Stockholm*, September 2003, p.p. 24-26. Available (04/02/2012) at: http://www.loa.istc.cnr.it/Publications.html

[Preuveneers et al. 2004] **PREUVENEERS**, D.,**VANDENBERGH**, J., **WAGELAAR**, D., **GEORGES**, A., **RIGOLE**, P., **CLERCKX**,T., **BERBERS**, Y., **CONINX**, K.,**JONCKERS**, V., and **DEBOSSCHERE**, K. "Towards an Extensible Context Ontology for Ambient Intelligence" In: *Ambient Intelligence,2004*. p.p. 148-159. http://dx.doi.org/10.1007/978-3-540-30473-9_15

[Raimond et al. 2007] **RAIMOND**,Y. and **ABDALLAH,** S. The event ontology, 2007. Available (04/02/2012) at: http://purl.org/NET/c4dm/event.owl

[Rainsford et al. 1999] **RAINSFORD**, C. and **RODDICK**,J. "Adding temporal semantics to association rules" In: *Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery - PKDD'99*. J. M. Zytkow and J. Rauch (Ed.) Springer, 1999. p.p. 504–509. http://dx.doi.org/10.1007/978-3-540-48247-5_65

[Randell et al. 1989] **RANDELL**,D.A. and **COHN**, A.G. "Modelling Topological and Metrical Properties in Physical Processes" In: *Principles of Representational Reasoning,* ed R Brachman et al, Morgan Kaufmann, Los Altos, 1989. Available (04/02/2012) at: ftp://agora.leeds.ac.uk/scs/doc/srg/KR89.ps

[Randell et al. 1992] **RANDELL**,D.A., **CUI**, Z. and **COHN**, A.G. "A spatial Logic Based on Regions and Connection" In: *Proceedings 3Rd International Conference on Knowledge Representation and Reasoning,* 1992. p.p.394-398. Available (04/02/2012) at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.7809&rep=rep1&type=pdf

[Reaz et al. 2007] **REAZ A.** et al. "Resource and Service Discovery in Large-Scale Multi-Domain Networks". In: Ieee Communications Surveys & Tutorials. 4Th Quarter 2007, Vol 9. Available at URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5473882&isnumber=5764312

[Reddy 2007] **REDDY**, B.S. Evidential reasoning for multimodal fusion in human computer interaction. MS Thesis, University of Waterloo, Canada, 2007. Available (04/02/2012) at: http://hdl.handle.net/10012/2674

[Reich 1994] **REICH**, A.J. "Intervals, Points, and Branching Time" In: *Proceedings of the TIME-94 International Workshop on Temporal Reasoning*, 1994. GOODWIN,S. D. and HAMILTON, H.J. (Ed.) Available (04/02/2012) at: http://www2.cs.uregina.ca/~temporal/time94/reich.pdf

[Reinecke et al. 2011] **REINECKE**, K. and **BERNSTEIN**, A. "Improving Performance, Perceived Usability, and Aesthetics with Culturally Adaptive User Interfaces" IN: ACM Transactions on Computer-Human Interaction, July 2011. Vol. 18. No. 2. http://doi.acm.org/10.1145/1970378.1970382

[Rich 1979] **RICH**, E. "User Modeling via Stereotypes" In: *Cognitive Science*, 1979. Vol.3. pp. 329-354.

[Roddick et al. 2005] **RODDICK**, J. F. and **MOONEY**, C. H. "Linear temporal sequences and their interpretation using midpoint relationships" In: *IEEE TKDE,* 2005. Vol.17. No.1. p.p.133–135. http://dx.doi.org/10.1109/TKDE.2005.12

[Rojas-Barahona et al. 2009] **ROJAS-BARAHONA,** L. M. and **GIORGINO**, T. "Adaptable dialog architecture and runtime engine (AdaRTE): A framework for rapid prototyping of health dialog systems". In: International Journal of Medical Informatics. April 2009. Vol. 78. pp. 56-68. http://dx.doi.org/10.1016/j.ijmedinf.2008.07.017

[Rousseau et al. 2004] **ROUSSEAU**, C., **BELLIK**, Y., **VERNIER**, F. and **BAZALGETTE**, D. "Architecture Framework For Output Multimodal Systems Design" In: *Proceedings of OZCHI'04,* Wollongong, Australia, 22-24 November 2004. http://www.ozchi.org/proceedings/2004/pdfs/ozchi2004-128.pdf

[Rousseau et al. 2006] **ROUSSEAU**, C., **BELLIK**, Y., **VERNIER**, F. and **BAZALGETTE**, D. "A Framework for the intelligent multimodal presentation of information" In: *Signal Processing*, Dec. 2006, Vol.86. No.12. pp.3696-3713. doi:10.1016/j.sigpro.2006.02.041

[Russell 1925] **RUSSELL**, B. The ABC of Relativity, 1925. 288 pages. ISBN-10: 0415154294

[Scherp et al. 2010] **SCHERP**, A., **FRANZ**, T., **SAATHOFF**, C. and **STAAB**, S. "F—A Model of Events based on the Foundational Ontology DOLCE+DnS Ultralite" In: *Proceedings of the fifth international conference on Knowledge capture -K-CAP09. ACM*, New York, NY, USA, p.p. 137-144. http://doi.acm.org/10.1145/1597735.1597760

[Schilit et al. 1994] **SCHILIT**, B.N., **HILBERT**, D.M. and **TREVOR,** J. "Context-aware communication" In: *Wireless Communications, IEEE*, 2002. Vol.9. No.5. p.p. 46-54. http://dx.doi.org/10.1109/MWC.2002.1043853

[Schmidt et al. 2001] **SCHMIDT,** A. and **VANLAERHOVEN**, K. "How to build smart appliances" In: *IEEE Personal Communications, Special Issue on Pervasive Computing,* August 2001. Vol.8. No.4. p.p.66–71 Available (04/02/2012) at: http://www.comp.lancs.ac.uk/~kristof/old/papers/PC_2001.pdf

[Scovanner et al. 2009] **SCOVANNER**, P. and **TAPPEN**, M. "Learning pedestrian dynamics from the real world" In: *Proc. International Conference on Computer Vision,* 2009, pp. 381–388. Available (04/02/2012) at: http://www.cs.ucf.edu/~mtappen/

[Searle 1969] **SEARLE**, J. R. "A Taxonomy of Illocutionary Acts". In: *Günderson, K. (Ed.), Language, Mind, and Knowledge*, Minneapolis. 1975, Vol.7.

[Searle 1995] **SEARLE**, J. R. The Construction of Social Reality. The Free Press, New York, 1995. ISBN-10: 0684831791

[Serrano et al. 2009] **SERRANO,**M., **NIGAY**,L., **LAWSON**,J-Y. L., **RAMSAY**, A., **MURRAY-SMITH**, R. and **DENEF**, S. "The openInterface framework: a tool for multimodal interaction" In: *CHI '08 extended abstracts on Human factors in computing systems -CHI EA08*. ACM, New York, NY, USA. p.p.3501-3506. http://doi.acm.org/10.1145/1358628.1358881

[Shaw et al. 2009] **SHAW**, R., **TRONCY**, R.and **HARDMAN**, L. "Linking Open Descriptions of Events" In: *Lecture Notes in Computer Science: The Semantic Web*. Springer Berlin-Heidelberg, 2009. Vol. 5926. p.p. 153-167. ISBN: 978-3-642-10870-9 http://dx.doi.org/10.1007/978-3-642-10871-6_11

[Singh et al. 2005] **SINGH**, S., **PURADKAR**, S., and **LEE**, Y. "Ubiquitous computing: Connecting pervasive computing through semantic web". In: *Information Systems and e- Business Management Journal.* Springer Berlin / Heidelberg 2005 http://dx.doi.org/10.1007/s10257-005-0003-8

[Sovis 2010] **SOVIS**, P., **KOHLAR**, F. and **SCHWENK**, J. "Security Analysis of OpenID" In: *Proceedings of Sicherheit 2010*, Lecture Notes in Informatics (LNI).

[Spence et al. 2004] **SPENCE**, C. and **DRIVER,** J. (Ed.) Crossmodal Space and Crossmodal Attention. Oxford University Press, 2004. 323 Pages.

[Strang et al. 2003] **STRANG**, T., **LINNHOFF-POPIEN**, C. and **FRANK**, K. "CoOL: A Context Ontology Language to enable Contextual Interoperability" In: *Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems-DAIS2003.* Paris, France, November 2003 Vol. 2893 of Lecture Notes in Computer Science (LNCS), Springer Verlag, pp. 236–247. Available (04/02/2012) at: http://elib.dlr.de/7338/

[Strang et al. 2004] **STRANG,** T. and **LINNHOFF-POPIEN**, C. "A Context Modeling Survey" In: *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management, Sixth International Conference on UbiComp'04.* 2004, Nottingham, England. Available (04/02/2012) at: http://itee.uq.edu.au/-pace/cw2004/Paper15.pdf

{SUN-JINI} **Sun** Microsystems JINI / RIVER Apache River (formerly JINI). Available at URL: http://river.apache.org/about.html

{SUN-JXTA 2001} **Sun** Microsystems Juxtapose JXTA Protocols Specification v2.0 October 16, 2007. v2.0 . Available at URL: http://jxta.kenai.com/Specifications/JXTAProtocols2_0.pdf

[Talantzis et al. 2006] **TALANTZIS**, F., **PNEVMATIKAKIS**, A. and **POLYMENAKOS**, L.C. "Real time audio-visual person tracking" In: *IEEE 8th Workshop on Multimedia Signal Processing,* 2006. Victoria, BC. pp. 243–247. http://dx.doi.org/10.1109/MMSP.2006.285306

{Thalmann et al. 1999} **THALMANN**, D. **MUSSE**, S.R. and **KALLMANN**, M. "Virtual Humans' Behaviour: Individuals, Groups, and Crowds" In: *Proceedings of the International Conference on Digital Media Futures, British Computer Society*, Bradford, UK. April 13-15, 1999. Available (06/02/1012) at: http://vrlab.epfl.ch/Publications/pdf/Thalmann_Musse_Kallmann_DMF_99.pdf

[Thorndike 1920] **THORNDIKE**, E.L. "Intelligence and its uses" In: Harper's Magazine, 1920. No.140. pp. 227-235.

[Town 2007] **TOWN**, C. "Multi-sensory and Multi-modal Fusion for Sentient Computing" In: *International Journal of Computer Vision,* 2007. Vol.71. No.2. pp. 235–253. http://dx.doi.org/10.1007/s11263-006-7834-8

[Traum et al. 2003] **TRAUM**, D.R. and **LARSSON**, S. "The Information state approach to dialog management". In: *Current and New Directions in Discourse and Dialog,* Kluwer, 2003.

[Troncy et al. 2009] **TRONCY**, R., **FIALHO**, A., **HARDMAN**, L. and **SAATHOFF**, C. "Experiencing Events through User-Generated Media" In: *Proceedings of the International Workshop on Consuming Linked Data - COLD'10, ISWC'10.* Vol. 665, Shanghai, China, November 7-11, 2010. Available (06/02/1012) at: http://www.eurecom.fr/~troncy/Publications/

[Ultsch 1996] **ULTSCH**, A. "Unification-based temporal grammar". Technical Report 37, Department of Mathematics and Computer Science, Philipps-University Marburg, Germany, 2004. Available (05/02/2012) at: https://www.mathematik.uni-marburg.de/forschung/publikationen/paper_info/bfi37.pdf

[Vainio et al. 2008] **VAINIO**,A., **VALTONEN**,M. and **VANHALA**, J. "Proactive Fuzzy Control and Adaptation Methods for Smart Homes" In: *IEEE Intelligent Systems.* March-April, 2008. Vol.23. No.2. pp.42-49. http://dx.doi.org/10.1109/MIS.2008.33

[Van Erp et al. 2005] **VAN ERP**, J.B.F., **VAN VEEN,** H.A.H.C . and **JANSEN**, C. "Waypoint navigation with a vibrotactile waist belt" In: *ACM Transactions on Applied Perception (TAP)*, 2005. Vol.2. No.2. pp. 106-117. doi:10.1145/1060581.1060585

[Van Hage et al. 2011] **VAN HAGE**, W.R., **MALAIS**, V., **SEGERS**, R., **HOLLINK**, L. and **SCHREIBER**, G. "Design and use of the Simple Event Model (SEM)" In: Web Semantics: Science, Services and Agents on the World Wide Web, 2011. Vol. 9. No. 2. Available (06/02/1012) at: http://www.websemanticsjournal.org/index.php/ps/article/view/190

[Van de Weghe et al. 2006] **VANDEWEGHE**, N., **COHN**, A. G., **DE TRE**, G. and **DE MAEYER**, P. "A Qualitative Trajectory Calculus as a Basis for Representing Moving Objects in Geographical Information Systems" In: *Control and Cybernetics*, 2006. Vol.35. Available (06/02/1012) at: http://www.comp.leeds.ac.uk/qsr/pub/ControlCyb05.pdf

[Varzi 2007] **VARZI**, A.C. "Spatial Reasoning and Ontology: Parts, Wholes, and Locations" In: *Handbook of Spatial Logics*, Berlin, Springer, 2007, Aiello, M., Pratt-Hartmann,I. E. and van Benthem,J. (Ed.) p.p. 945-1038. Available (06/02/2012) at: http://www.columbia.edu/~av72/recent.html

[Vazquez et al. 2007] **VAZQUEZ**, J. I., **LOPEZ DE IPIÑA**, D. and **SEDANO**. I. "Soam: A web-powered architecture for designing and deploying pervasive semantic devices" . In: *International Journal of Web Information Systems*, 2007 http://dx.doi.org/10.1108/17440080780000301

[Vieu 1997] **VIEU**, L. "Spatial representation and reasoning in artificial intelligence" In: *Spatial and Temporal Reasoning,* 1997. Kluwer, Dordrecht, Netherlands, p.p. 5–41. http://dx.doi.org/10.1007/978-0-585-28322-7_1

[Villafane 1999] **VILLAFANE**, R., **HUA**,K. A., **TRAN**,D. and **MAULIK**,B. "Mining interval time series" In: *Proceedings of the 1st International Conference on Data Warehousing and Knowledge Discovery -DaWaK'99*.Springer, 1999. p.p. 318–330. http://dx.doi.org/10.1007/3-540-48298-9_34

[Vinciarelli et al. 2009] **VINCIARELLI**, A., **PANTIC**, M. and **BOURLARD**, H. "Social signal processing: Survey of an emerging domain" In: *Image and Vision Computing*, November, 2009. Vol. 27. No.12. p.p.1743–1759 http://dx.doi.org/10.1016/j.imavis.2008.11.007

[Wang et al. 2004] **WANG**,X.H., **ZHANG**, D.Q., **GU,**T. and **PUNG**,H.K. "Ontology-Based Context Modeling and Reasoning using OWL" In : *Context Modeling and Reasoning Workshop*, pp. 18–22, Mar 2004. http://dx.doi.org/10.1109/PERCOMW.2004.1276898

[Walters et al. 2005a] **WALTERS**, M.L., **DAUTENHAHN**, K., **BOEKHORST**, R., **KOAY,** K.L., **KAOURI**, C., **WOODS**, S., **NEHANIV**, C., **LEE**, D. and **WERRY**, I. "The influence of subjects' personality traits on personal spatial zones in a human-robot interaction experiment" In: *IEEE International Workshop on Robot and Human Interactive Communication*, 2005. ROMAN 2005. http://dx.doi.org/10.1109/ROMAN.2005.1513803

[Walters et al. 2005b] **WALTERS**, M.L., **DAUTENHAHN**, K., **KOAY**, K.L., **BOEKHORST**, R., **NEHANIV**, C., **WERRY**,I. and **LEE**, D. "Close encounters: Spatial distances between people and a robot of mechanistic appearance" in Proceedings of the IEEE-RAS International Conference on Humanoid Robots, Tsukuba, Japan, 2005, pp. 450–455. http://dx.doi.org/10.1109/ICHR.2005.1573608

[Weizenbaum 1966] **WEIZENBAUM**, J. "ELIZA — a computer program for the study of natural language communication between man and machine" In: *Communications of the ACM*, jan. 1966. Vol. 9 No.1, pp:36-45. http://dx.doi.org/10.1145/365153.365168

[Wooldridge 2002] **WOOLDRIDGE**, M. An Introduction to MultiAgent Systems. John Wiley & Sons Ltd, 2002, paperback, 366 pages, ISBN 0-471-49691-X.

[Zhihong et al. 2009] **ZHIHONG,** Z., **PANTIC**, M., **ROISMAN**, G.I. and **HUANG**, T.S. "A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Jan. 2009. Vol.31. No.1. pp.39-58. http://dx.doi.org/10.1109/TPAMI.2008.52

2 Standards and Open Standards.

[CENELEC-CANOpen 2002] **CANOpen Electronic Data Sheet.** Text files on ASCII format for CANOpen nodes into networks. 2002. Available (04/02/2012) at: http://can-cia.org/

[DoJ-JXDM 2005] **Global Justice XML Data** Model. Available (04/02/2012) at: http://it.ojp.gov/jxdm/

[ECHONET 2001] **ECHONET** Energy Conservation and Homecare NETwork. http://www.echonet.gr.jp/

[ECHONET-DeviceObjects 2002] **ECHONET** Consortium. Detailed stipulation for Echonet Device Objects Available at URL: http://www.echonet.gr.jp/english/spec/pdf/spec_v1e_appendix.pdf

[ECMA-262] **ECMA** International. ECMAScript Language Specification, Third Edition. December 1999. URL: http://www.ecma-international.org/publications/standards/Ecma-262.htm

[FIPA-DeviceOnt 2002] **FIPA Device Ontology Specification**, 06/12/2002. Available (04/02/2012) at: http://www.fipa.org/specs/fipa00091/index.html

[ICANN-IANA 1998] **IANA** Domain Name Service ICCAN - Internet Assigned Numbers Authority (IANA). Available at URL: http://www.iana.org/

[IDA-FDCML 2001] **FDCML. Field Device Configuration Markup Language 2.0.** 2001. Available (04/02/2012) at: http://www.fdcml.org/

[IEC-EDDL 2006] **EDDL. Electronic Device Description Language.** 2006. Available (04/02/2012) at: http://www.eddl.org

[IEC-GSDML 2003] **GSDML. Generic Station Description Markup Language.** 2003. Available (04/02/2012) at: http://www.profibus.com/nc/downloads/downloads/gsdml-specification-for-profinet-io/display/

[IEEE-1451 1993] **IEEE Standard for a Smart Transducer Interface for Sensors and Actuators** http://ieeexplore.ieee.org/xpl/standards.jsp?findtitle=1451&letter=1451&opentree=on

[IETF-RFC2608 1999] **GUTTMAN,** E. et al. SLP. Service Location Protocol. RFC 2608 Version 2,1999. URL: http://www.ietf.org/rfc/rfc2608.txt

[IETF-RFC4627 2006] **The application/json Media Type for JavaScript Object Notation (JSON)** Available at: http://www.ietf.org/rfc/rfc4627.txt

[ISO-ASN.1 1984] **LARMOUTH**,J., **STEEDMAN**,D. and **WHITE**, J.E. ASN.1 International Telecommunication Union. ISO/IEC 8824-1 > 8824-4. Available at URL: http://www.itu.int/ITU-T/asn1/introduction/index.htm

[ISO-13407 1999] **ISO 13407:1999**. Human-centered design processes for interactive systems. Available (04/02/2012) at: http://www.iso.org/iso/catalogue_detail.htm?csnumber=21197

[ISO-16262 1997] Ecma International in the ECMA-262 specification and ISO/IEC 16262.

[ISO-MPEG21 2002] **ISO/IEC 21000 - Multimedia framework** http://www.mpegif.org/resources.php#section42

[ISO-MPEGV 2011] **Media context and control.** (ISO/IEC 23005) More info at: http://www.itscj.ipsj.or.jp/sc29/29w42911.htm#MPEG-V

[MODBUS 1979] **MODBUS Device Directory. 1979.** Available (04/02/2012) at: http://www.modbus.org/devices.php

[OASIS-DPWS 2009] **Devices Profile for Web Services 1.1.** 30/06/2009. Available (04/02/2012) at: http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01

[OASIS-WSDiscovery 2009] **WS-Discovery. Web Services Dynamic Discovery.** 01/07/2009. Available (04/02/2012) at: http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01

[OASIS-XCBF 2011] **The OASIS XML Common Biometric Format.** Available (04/02/2012) at: http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xcbf

[OMA-UAProf 2001] **UAProf.** The User Agent Profile specification. Available (04/02/2012) at: http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf

[OMG-UML 2007] **Unified Modeling Language.** http://www.uml.org/

[UPnP-Arch 2008] **UPnP™ Device Architecture 1.1**. 15/10/2008. Available (04/02/2012) at: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf

[WP2-FIDIS 2005] Future of IDentity in the Information Society Available (04/02/2012) at: http://www.fidis.net

[W3C-ARIA 2010] **CRAIG,** J et al. Accessible Rich Internet Applications WAI-ARIA version 1.0. W3C Working Draft, 16 September 2010. Available at URL: http://www.w3.org/TR/wai-aria/

[W3C-CC/PP 2010] **CC/PP.** Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0 Available (04/02/2012) at: http://www.w3.org/TR/CCPP-struct-vocab2/

[W3C-EMOTIONML-USE 2005] **SCHRÖDER**,M., **ZOVATO**, E., **PIRKER**,H. **PETER**,C., **BURKHARDT**,F. Emotion Incubator Group Use Cases Available at URL: http://www.w3.org/2005/Incubator/emotion/XGR-emotion/#AppendixUseCases

[W3C-EMMA 2009]	**Johnston**,M et al. EMMA: Extensible MultiModal Annotation markup language. 10 February 2009. W3C Recommendation. URL: http://www.w3.org/TR/2009/REC-emma-20090210/
[W3C-DCO 2007]	**DCO.** Delivery Context Ontology. Working Draft, December 2007. Available (04/02/2012) at: http://www.w3.org/TR/dcontology/
[W3C-DDR 2008]	**DDR - Device Description Repository.** Recommendation, December 2008. Available (04/02/2012) at: http://www.w3.org/TR/2008/REC-DDR-Simple-API-20081205/
[W3C-HTML5 2012]	Hypertext Markup Language. Available (04/02/2012) at: http://www.w3.org/TR/html5/
[W3C-EmotionML 2011]	**Emotion Markup Language (EmotionML) 1.0** http://www.w3.org/TR/emotionml/
[W3C-IndieUI 2011]	**Indie UI Working Group Activity** (event models for Application Programming Interfaces) Available (04/02/2012) at: http://www.w3.org/2011/11/indie-ui-charter
[W3C-MCAPI 2010]	**Media Capture API.** Part of the device API Specification. Available (04/02/2012) at: http://www.w3.org/TR/media-capture-api/
[W3C-MMI 2011]	**Multimodal Interaction Framework** http://www.w3.org/TR/mmi-framework/
[W3C-OWLS 2004]	**OWL-S: Semantic Markup for Web Services** Available at: http://www.w3.org/Submission/OWL-S/
[W3C-SAWSDL 2007]	**Semantic Annotations for WSDL and XML Schema** Available at: http://www.w3.org/TR/sawsdl/
[W3C-SOAP 2003]	Simple Object Access Protocol http://www.w3.org/2002/07/soap-translation/soap12-part0.html
[W3C-RDF 2004]	**Resource Description Framework (RDF)** http://www.w3.org/standards/techs/rdf#w3c_all
[W3C-RDFS 2004]	**Resource Description Framework Schema (RDFS)** **http://www.w3.org/TR/rdf-schema/**
[W3C-WEBINTENTS 2012]	**KINLAN**, P et al. **Web Intent**s. Available at URL: http://www.w3.org/TR/2012/WD-web-intents-20120626/
[W3C-WSDL 2001]	**Web Services Description Language 1.1.** 15/03/2001. Available (04/02/2012) at: http://www.w3.org/TR/wsdl
[W3C-WSPolicy 2006]	**Web Services Policy 1.2.** 25/04/2006 Available (04/02/2012) at: http://schemas.xmlsoap.org/ws/2004/09/policy/
[W3C-XML 2004]	**BRAY**,T., **PAOLI**,J.,**SPERBERG-MCQUEEN**, C.M., **MALER**, E., **COWAN**,J., **YERGEAU**, F. XML Extensible Markup Language, XML 1.1, 4 February 2004. URL: http://www.w3.org/TR/xml11/
[W3C-XMLSchema 2011]	**XML Schema Definition Language 1.1.** 21/07/2011 Available (04/02/2012) at: http://www.w3.org/TR/xmlschema11-1/
[WP2-FIDIS 2005]	**Future of Identity in the Information Society** Available (04/02/2012) at: http://www.fidis.net/
[WURFL 2011]	**Wireless Universal Resource File 2.1.** June 2011. Available (04/02/2012) at: http://wurfl.sourceforge.net/

LISTS

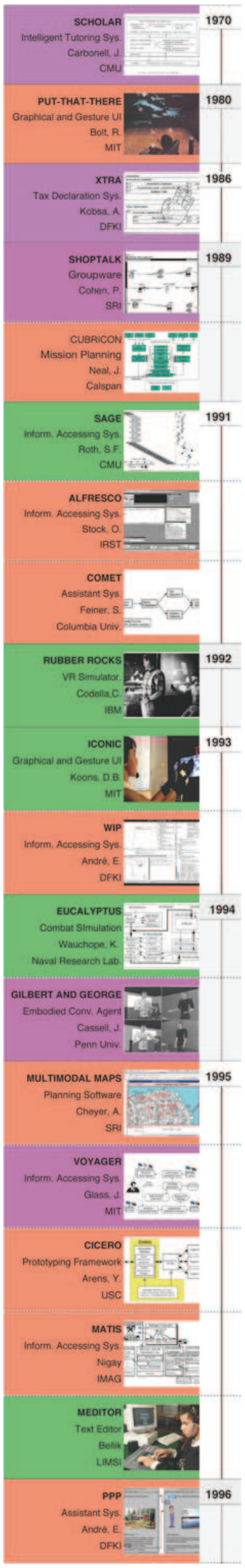
1 List of Figures

2 List of Tables

2 List of Definitions and Axioms

APPENDIX

1 Multimodal Architectures Timeline



Year	Project	Field	Lab	Application Type	Biblio
1970	SCHOLAR	Learning	CMU	Intelligent Tutoring Sys.	CARBONELL, Jaime. "Mixed-In
1980	PUT-THAT-THERE	GUI Experiment	MIT	Graphical and Gesture U	BOLT, Richard A. " 'Put-that-ther
1986	XTRA	Citizenship	DFKI	Tax Declaration Sys.	Allgayer, J., K. Harbusch, A. Kol
1989	SHOPTALK	Manufacturing	SRI	Groupware	P. R. Cohen, M. Dalrymple, D. E
1989	CUBRICON	Military	Calspan	Mission Planning	Jeannette G. Neal and Stuart C
1991	SAGE	Multimedia Repr	CMU	Inform. Accessing Sys.	Roth, S.F.; Mattis, J.; Automat
1991	ALFRESCO	Art	IRST	Inform. Accessing Sys.	Oliviero Stock: Natural Languag
1991	COMET	Assistant	Columbia Ur	Assistant Sys.	Steven K. Feiner and Kathleen
1992	RUBBER ROCKS	Entertainment	IBM	VR Simulator.	Christopher Codella, Reza Jalili
1993	ICONIC	GUI Experiment	MIT	Graphical and Gesture U	David B. Koons and Carlton J. S
1993	WIP	Multimedia Repr	DFKI	Inform. Accessing Sys.	Elisabeth André, Wolfgang Fink
1994	EUCALYPTUS	Military	Naval Resea	Combat SImulation	Wauchope, K. Eucalyptus: Integ
1994	GILBERT AND GEORG	Assistant	Penn Univ.	Embodied Conv. Agent	CASSELL. MODELING THE IN
1995	MULTIMODAL MAPS	Travel Assistant	SRI	Planning Software	CHEYER A. & JULIA L. Multimo
1995	VOYAGER	Geo-Location	MIT	Inform. Accessing Sys.	James Glass, Giovanni Flammi
1995	CICERO	Software Desigr	USC	Prototyping Framework	Yigal Arens and Eduard Hovy. 1
1995	MATIS	Travel Assistant	IMAG	Inform. Accessing Sys.	Laurence Nigay and Joëlle Cou
1995	MEDITOR	Assistant	LIMSI	Text Editor	BELLIK, Y. Interfaces Multimod
1996	PPP	Travel Assistant	DFKI	Assistant Sys.	Elisabeth André;, Thomas Rist,
1996	GANDALF	Learning	MIT	Embodied Conv. Agent	Thórisson, Kristinn Rúnar. Com
1996	MAGIC	Medical	Columbia Ur	Inform. Accessing Sys.	Mukesh Dalal, Steven Feiner, K
1996	FLASH	Software Desigr	Adobe	Media Player	http://blogs.adobe.com/actionsc
1996	VISUALMAN	GUI Experiment	Hangzhou U	Voice and Eye Gaze UI	Jian Wang. Integration model of
1996	JEANIE	Calendar	CMU	Assistant Sys.	Minh Tue Vo and Cindy Wood. I
1997	QUICKSET	Geo-Location	Oregon Grac	Inform. Accessing Sys.	Philip R. Cohen, Michael Johns
1997	CARTOON	Geo-Location	LIMSI	Inform. Accessing Sys.	MARTIN, J-C. Towards "intellige
1998	MASK	Geo-Location	LIMSI	Inform. Accessing Sys.	A Spoken Language System Fc
1998	AUGUST	Art	KTH	Embodied Conv. Agent	Joakim Gustafson, Magnus Lur
1998	DENK	Assistant	Tilburg/Eindh	Assistant Sys.	Bunt, Harry;Ahn, René; Beun, F
1998	AVANTI	Browser	Eu Project	Inform. Accessing Sys.	C. Stephanidis , A. Paramythis ,
1998	DIGITAL SMARTKIOSK	Kiosk	Cambridge F	Assistant Sys.	Andrew D. Christian and Brian L
1999	COSMO	Learning	NCS Univ.	Intelligent Tutoring Sys.	W. Lewis Johnson and W. Ricke
1999	STEVE	Learning	NCS Univ.	Embodied Conv. Agent	W. Lewis Johnson , Jeff W. Rick
1999	WILL	Meeting Assista	FXPAL	Embodied Conv. Agent	Prevost, S., Hodgson, P., Cook,
1999	REA	Real-State	MIT	Embodied Conv. Agent	CASSELL, J. "Embodied Conve
2000	ADAPT	Real-State	KTH	Embodied Conv. Agent	Gustafson, Joakim. Developing
2000	MIPAD	Assistant	Microsoft	Inform. Accessing Sys.	X. Huang, A. Acero, C. Chelba,
2000	PDAMVPQ	Corporate	AT&T	Inform. Accessing Sys.	Michael Johnston and Srinivas I
2000	GPAC	Software Desigr	Telecom Par	Media Player	LEFEUVRE. GPAC, Open Sou
2001	MATCH	Travel Assistant	AT&T	Inform. Accessing Sys.	Michael Johnston, Srinivas Ban
2001	COLLAGEN	Travel	MERL	Intelligent Tutoring Sys.	Charles Rich, Candace L. Sidne
2001	MRE	Learning	USC	VR Mission Planning	W Swartout and R Hill and J Gr
2001	GRETA	Assistant	Roma Univ.	Embodied Conv. Agent	PELACHAUD, Catherine, Caro
2002	ECHART	Medical	AT&T	Healthcare Rec. Annotati	US Patent number: 7225131
2002	MACK	Kiosk	MIT	Mixed Reality Sys.	Tom Stocky and Justine Cassel
2002	RASA	Military	OGI	Mission Planning	McGee, D.R., et al. "Comparing
2002	SGIM	GUI Experiment	Bielefeld Uni	Speech, 3D and Gesture	Marc Erich Latoschik. 2002. De
2002	INTERACT	Geo-Location	NTA	Inform. Accessing Sys.	Kristiina Jokinen and Antti Kerm
2002	EMBASSI	Entertainment	Fraunhofer	Assistant Sys.	Thoma Heider and Thomas Kir
2002	IMBUILDER	Software Desigr	University of	Prototyping Framework	Marie-Luce Bourguet. A Toolkit f
2002	IMAP	Emergency Res	Penn Univ.	Assistant Sys.	Krahnstoever, N.; Schapira, E.;
2002	VICO	Drive Assistant	NISLab,	Assistant Sys.	Bernsen, Niels Ole; Dybkjr, Laila
2003	GALATEA	Software Desigr	ISTC	Prototyping Framework	Nitta. Activities of Interactive Spe
2003	SMARTKOM	Kiosk	DFKI	Inform. Accessing Sys.	Herzog, Gerd and Reithinger, N
2003	PETSHOP	CAD	LIIHS-IRIT	Prototyping Framework	Rémi Bastide, David Navarre, F
2003	FLATSCAPE FW	CAD	Rutgers Univ	Prototyping Framework	Frans Flippo, Allen Krebs, and I
2003	CROSSWEAVER	CAD	Washington	Prototyping Framework	Anoop K. Sinha and James A. L

2003	MAGICSTER	Assistant	Eu Project	Mixed Reality Sys.	Catherine Pelachaud, Valeria Carofiglio, B
2003	MERCURY	Assistant	DSTC	Ambient Intelligence	Karen Henricksen. A framework for Conte
2003	FINGER-POINTER	GUI Experiment	Phillips	Graphical and Gesture UI	US Patent number: 6600475
2004	COMIC	CAD	ICCS	Groupware	Mary Ellen Foster, Michael White, Andrea
2004	MATCHKIOSK	Kiosk	AT&T	Inform. Accessing Sys.	Michael Johnston and Srinivas Bangalore
2004	NICE	Entertainment	EU Project	Game	J. Gustafson, L. Bell, J. Boye, A. Lindströn
2004	DPD	CAD	MindGrip	Prototyping Framework	A. D. Milota. 2004. Modality fusion for grap
2004	ICARE	CAD	TIMC-IMAG	Prototyping Framework	Bouchet, J. Ingénierie de l'interaction multi
2004	INTUITION	Military	TIMC-IMAG	VR Simulator	Bouchet, J. Ingénierie de l'interaction multi
2004	M2-M4 MACACO	Entertainment	MIT	Robotics	ARSENIO, A. M. "Towards an Embodied
2004	OZONE	Geo-Location	LORIA	Inform. Accessing Sys.	Bertrand Gaiffe, Frédéric Landragin, Matth
2004	MIRAGE	GUI Experiment	Columbia Univ.	Inform. Accessing Sys.	Thorisson, K., Benko, H., Abramov, D., Arr
2004	SMART HOMES	Medical	Telecom ParisTec	Ambient Intelligence	Mohamed Ali Feki, Stéphane Renouard,
2005	SMARTWEB	Entertainment	DFKI	Inform. Accessing Sys.	Norbert Reithinger, and Daniel Sonntag. A
2005	FATIMA				
2005	FAME FW	CAD	LaSIGE	Prototyping Framework	Carlos Duarte and Lu\&\#237;s Carri\&\#2
2005	M4:MM MEETING MNGR.	Meeting Assistant	IDIAP	Ambient Intelligence	MCCOWAN, I., GATICA-PEREZ, D., BE
2006	RIA	Real-State	IBM	Inform. Accessing Sys.	Michelle X. Zhou Keith Houck Shimei Pan
2006	ARCHIVUS	Meeting Assistant	EPFL	Inform. Accessing Sys.	Miroslav Melichar and Pavel Cenek. 2006
2006	FAME	Meeting Assistant	EU Project	Ambient Intelligence	Florian Metze, Petra Gieselmann, Hartwig
2006	MIMUS	Assistant	Sevilla Univ.	Mixed Reality Sys.	Perez, G.; Amores, G.; Manchon, P.; , "A
2006	PERM	Medical	TIMC-IMAG	Computed-Assisted Surgery	Benoit Mansoux, Laurence Nigay, Jocely
2006	SAMMIE	In-Car Entertainment	DFKI	Inform. Accessing Sys.	Norbert Pfleger, Jan Scheh. Development
2006	ELOQUENCE	Military	DGA	VR Mission Planning	ROUSSEAU, C., BELLIK, Y., VERNIER,
2006	SAL				
2006	SAIBA				
2006	OMISCID	Software Design	INRIA	Middleware	Barraquand, R, Vaufreydaz, D., Emonet,
2006	SMARTOFFICE	Assistant	INRIA	Ambient Intelligence	CROWLEY,J., BRDICZKA, O. and REIG
2007	CONQUEST	Meeting Assistant	CMU	Assistant Sys.	ConQuest: An open-source dialog system
2008	HEPHAISTK	Software Design	University of Fribo	Prototyping Framework	Dumas, B., Lalanne, D., and Ingold, R. 20
2008	K-SPACE	Media Annotation Toc	EU Project	Multimedia Analysis	SHAW, R., TRONCY, R.and HARDMAN,
2008	WE KNOW IT	Emergency Respons	CFR&T	Multimedia Analysis	SCHERP, A., FRANZ, T., SAATHOFF, C.
2009	SSI	Software Design	Ausgburg Univ.	Prototyping Framework	Johannes Wagner, Elisabeth André, Fran
2009	OPENINTERFACE	Software Design	EU Project	Prototyping Framework	SERRANO,M., NIGAY,L., LAWSON,J-Y.
2009	SQUIDY	Software Design	Konstanz Univ.	Prototyping Framework	Werner A. König, Roman Rädle, and Hara
2010	CHLOE@UNIVERSITY	Assistant	EU Project	Mixed Reality Sys.	Achille Peternier, Xavier Righetti, Mathieu
2010	MULTIMODAL KIOSK	Kiosk	Microsoft	Assistant Sys.	BOHUS, D. and HORVITZ, E. "On the Ch

This timeline covering 100 multimodal projects is the basis for the selection of the 16 architectures analyzed empirically in our state of the art. It was produced as an effort to classify the different approaches according to our conceptual model, trying to detect the main orientation of research on each project. It also allow us to see some trends over time in the multimodal field research. Every paper was studied with the goal of finding information about the architecture implemented or generated by the solution. Functional blocks, responsibilities and context issues where explored in order to select the more representative and complete proposals.

2 Implementation Report

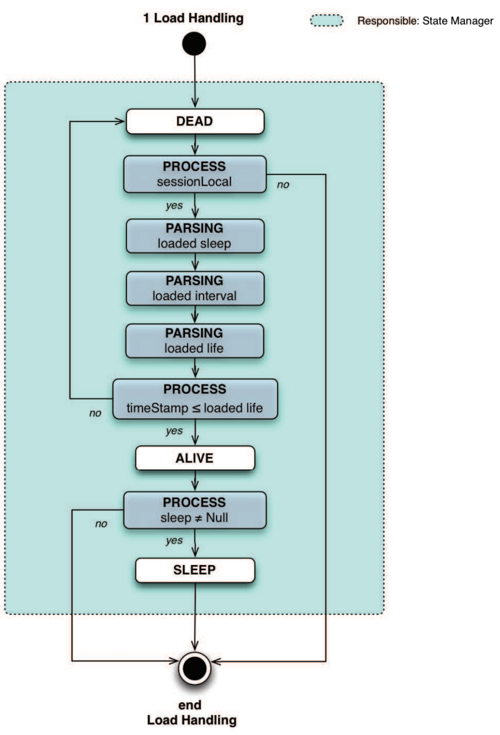
The following is the detailed specification of the implementation of the MMILib for Modality Components:

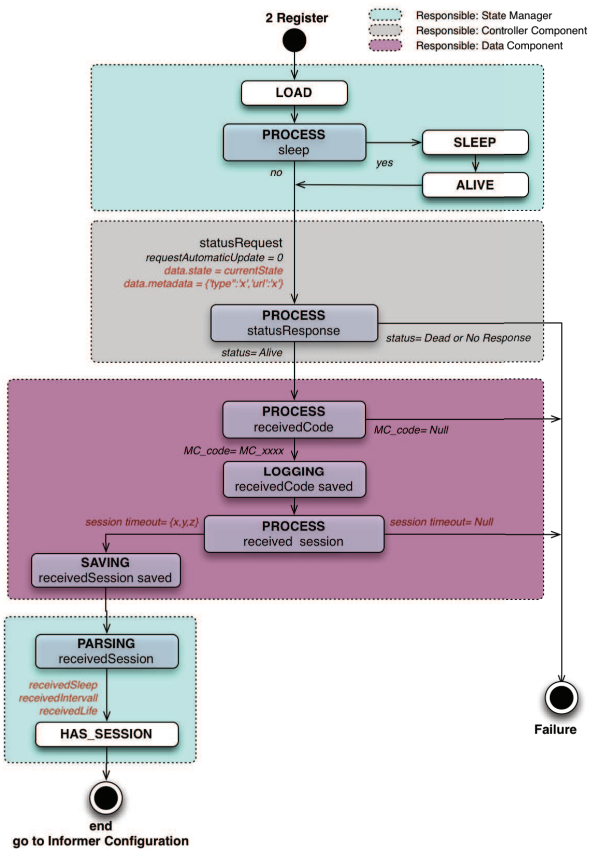
Common	Modality Components communicate with the Interaction manager via asynchronous events.	OK
	Constituents must be able to handle events that are delivered to them asynchronously.	OK
	All lifecycle events must use a common basic concept of 'context'.	OK
	A context must represent a single extended interaction with zero or more users across one or more modality components.	OK
	A context should cover the longest period of interaction over which it would make sense for components to store information.	OK
	All events relating to a given interaction must use the same context URI.	OK
	Any two events that use different context URIs must be interpreted as parts of unrelated interactions.	OK
Source	The Source attribute must be a URI representing the address of the sender of the event.	OK
Target	The Target attribute must be a URI representing the address of the destination of the event.	OK
RequestID	The RequestID attribute must be an identifier that is unique within the given context for each Request/Response pair.	OK
	For any Request/Response event pair, the RequestID in the Response event must match the RequestID specified in the Request event.	OK
Status	The Status attribute must be either 'success' or 'failure'.	OK
	The Response event of a Request/Response pair must use the Status field to report whether it succeeded in carrying out the request.	OK
StatusInfo	The Response event of a Request/Response pair MAY use the StatusInfo field to provide additional status information.	OK
	All constituents MUST be able to process Life-Cycle events that use the StatusInfo field to provide additional status information.	OK
Data	Any event MAY use the Data field to contain arbitrary data.	OK
	All constituents MUST be able to process Life-Cycle events that use the Data field to provide arbitrary data.	OK
NewContext	A modality component MAY send a NewContextRequest event to the interaction manager to request that a new context be created.	OK
Prepare	The IM MAY send a PrepareRequest to allow the Modality Components to pre-load markup and prepare to run.	OK
	The PrepareRequest event MAY contain a ContentURL field with the URL of the content that the Modality Component SHOULD prepare to execute.	OK
	The PrepareRequest event MAY contain a Content field with inline markup that the Modality Component SHOULD prepare to execute.	OK
	The IM MAY leave both contentURL and content fields of a PrepareRequest event empty.	OK
	The IM MAY include a value in the ContentURL or Content field of the StartRequest event.	OK
	The IM MAY use the same context value in multiple StartRequest events when it wishes to execute multiple instances of markup in the same context.	NO
	The StartRequest event MAY contain a ContentURL field with the URL of the content that the Modality Component MUST attempt to execute	OK
	The StartRequest event MAY contain a Content field with inline markup that the Modality Component MUST attempt to execute.	OK
	The IM MAY leave both contentURL and content fields of a StartRequest event empty.	OK
	The IM MAY send a CancelRequest to stop processing in the Modality Component.	OK
	The IM MAY send a PauseRequest to suspend processing by the Modality Component.	OK
	PauseRequest MUST contain an Immediate field which is a boolean value.	OK
Resume	The IM MAY send the ResumeRequest to resume processing that was paused by a previous PauseRequest.	OK
ExtensionNotif	The ExtensionNotification event MAY be generated by the IM to encapsulate application-specific events.	OK
	The ExtensionNotification event MAY be generated by the Modality Component to encapsulate application-specific events.	OK
	All constituents MUST be able to process ExtensionNotification events.	OK
Status	The IM MAY send a StatusRequest event to a MC to inquire about the status of a specific user interaction (context) or the status of the MC itself.	OK
	The recipient of a StatusRequest event MUST respond with a StatusResponse event.	OK
	A MC MAY send a StatusRequest event to the IM to inquire about the status of a specific user interaction (context) or the status of the IM itself.	OK
	The sender of a StartRequest event MAY use the Context field to specify the context for which the status is requested.	OK
	The StatusRequest event MUST contain a RequestAutomaticUpdate field which is a boolean value.	OK
	If the Context field is present in a StatusRequest message, the recipient MUST respond with a StatusResponse message indicating the status of the specified context.	OK
	If the Context field is not present in a StatusRequest message, the recipient MUST send a StatusResponse message indicating the status of the underlying server.	OK
	If the RequestAutomaticUpdate field of a StatusRequest message is 'true', the recipient SHOULD send periodic StatusResponse messages without waiting for an	OK
	If the RequestAutomaticUpdate field of a StatusRequest message is 'false', the recipient SHOULD send one and only one StatusResponse message in response to	OK
	The sender of a StartResponse event MAY use the Context field to specify the context for which the status is being returned.	OK
	The StatusResponse event MUST contain an AutomaticUpdate field which is a boolean value.	OK
	If the AutomaticUpdate field of a StatusResponse message is 'true', the sender MUST keep sending StatusResponse messages in the future without waiting for	OK
	If the AutomaticUpdate field of a StatusResponse message is 'false', the sender MUST wait for a subsequent StatusRequest message before sending another	OK
	If the Context field is present in a StatusResponse event, the response MUST represent the status of the specified context.	OK
	The Status field of a StatusResponse message is an enumeration of 'Alive', 'Dead', 'Unknown'	OK
	If the StatusResponse message specifies a context which is still active and capable of handling new life cycle events, the sender MUST set the Status field to 'Alive'	OK
	If the StatusResponse message specifies a context which has terminated or is otherwise unable to process new life cycle events, the sender MUST set the Status to	OK
	If the StatusResponse message doesn't provide a Context field, and the sender is able to create new contexts, it MUST set the Status to 'Alive'.	OK
	If the StatusResponse message doesn't provide a Context field, the sender MUST set the Status to 'Unknown'.	OK
	If the StatusResponse message doesn't provide a Context field, and the sender is unable to create new contexts, it MUST set the Status to 'Dead'.	OK
	All Modality Components must support the basic life-cycle events.	OK
	Modality Components MUST return a PrepareResponse event in response to a PrepareRequest event.	OK
	Modality Components that return a PrepareResponse event with Status of 'Success' SHOULD be ready to run with close to 0 delay upon receipt of the StartRequest.	OK

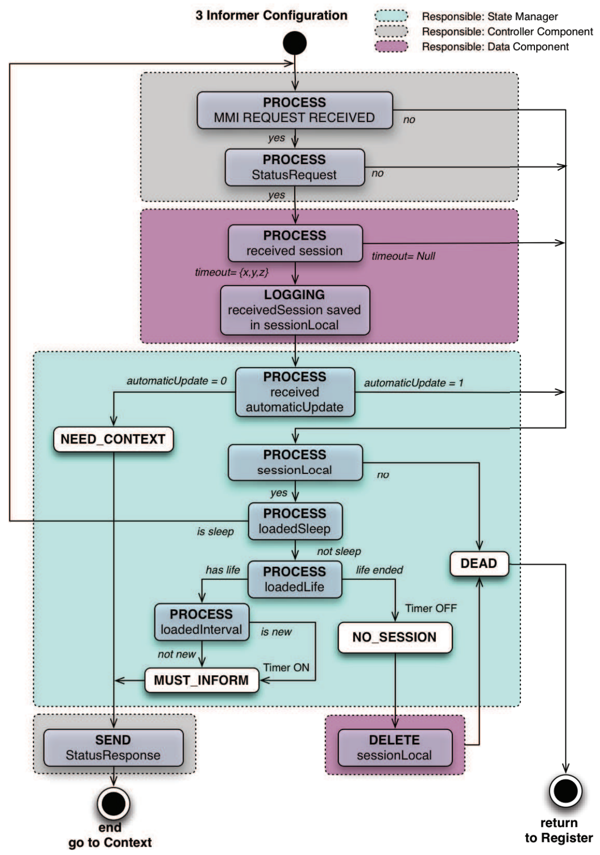
	When it receives multiple PrepareRequests, the Modality Component SHOULD prepare to run any of the specified content.	OK
	If both contentURL and content of a PrepareRequest are empty, the Modality Component MUST revert to its default behavior.	OK
	If a MC receives a PrepareRequest containing a new context (without a previous NewContextRequest/Response exchange), it MUST accept the new context and	OK
	The Modality Component MUST return a StartResponse event in response to a StartRequest event.	OK
	If the IM includes a value in the ContentURL or Content field of the StartRequest event, the Modality Component MUST use this value.	OK
	If a Modality Component receives a new StartRequest while it is executing a previous one, it MUST either cease execution of the previous StartRequest and begin	OK
	If the IM leaves both contentURL and content of a StartRequest event empty, the Modality Component MUST run the content specified in the most recent	OK
	If the Modality Component did not receive a PrepareRequest event prior to receiving a StartRequest event with empty Content and ContentURL fields, it MUST revert	OK
DoneNotificati	If a modality component finishes processing it MUST send a DoneNotification	OK
Cancel	A Modality Component MUST return a CancelResponse event in response to a CancelRequest event.	OK
	A Modality Component that receives a CancelRequest event MUST stop processing and then MUST return a CancelResponse.	OK
	If the value of the Immediate field in a CancelRequest event is 'true', the Modality Component SHOULD stop processing for the specified context immediately.	OK
	If the value of the Immediate field in a CancelRequest event is 'false', the Modality Component SHOULD stop processing for the specified context gracefully.	OK
	CancelResponse MUST be sent by the Modality Component as a response to the CancelRequest event	OK
	CancelResponse MUST contain a Status field.	OK
Pause	Modality Components MUST return a PauseResponse once they have paused, or once they determine that they will be unable to pause.	OK
	If the value of the Immediate field in a PauseRequest event is 'true', the Modality Component SHOULD pause processing for the specified context immediately.	OK
	If the value of the Immediate field in a PauseRequest event is 'false', the Modality Component SHOULD pause processing for the specified context gracefully.	OK
	Implementations that have received a ResumeRequest event even though they haven't paused MUST return a ResumeResponse with a Status of 'success'.	OK
	The 'Status' of a ResumeResponse event MUST be 'success' if the implementation has succeeded in resuming processing and MUST be failure otherwise	OK
	Implementations that have paused MUST attempt to resume processing upon receipt of this event and MUST return a ResumeResponse afterwards.	OK
	A MC MUST send a ClearContextResponse event in response to a ClearContextRequest event, even if doesn't take any particular action.	OK

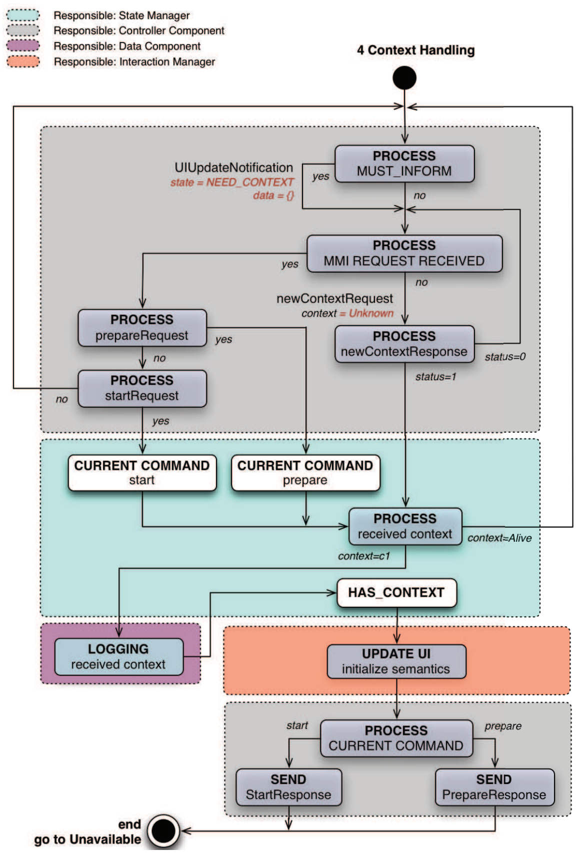
3 State Diagrams for Registration

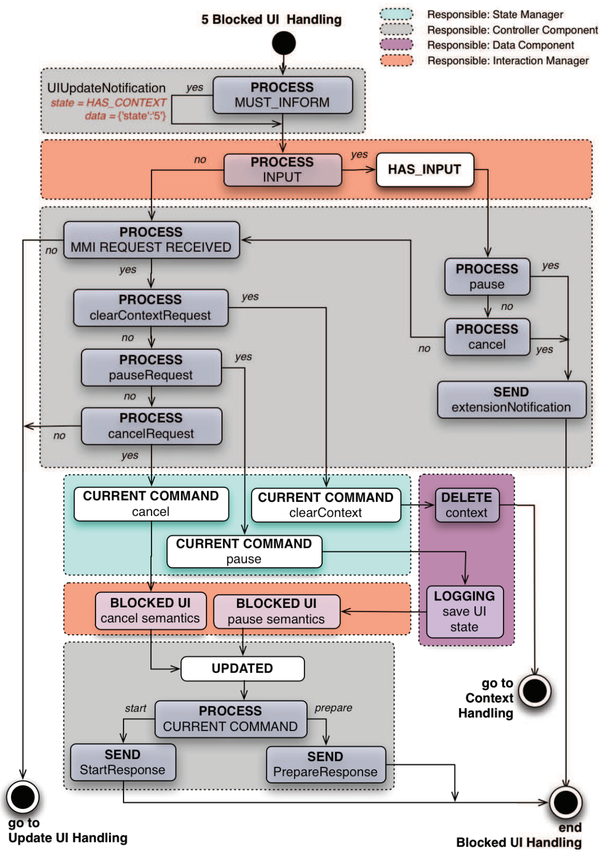
The following are the state diagrams used for the development of the push mechanism according with the MMI recommendation and handling state management (by the use of the timeout structures) and semantic annotations:

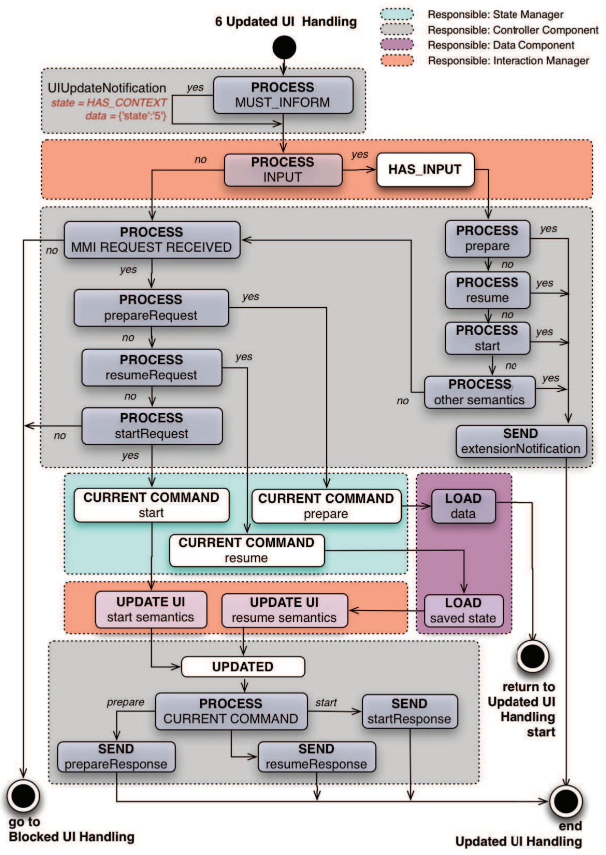






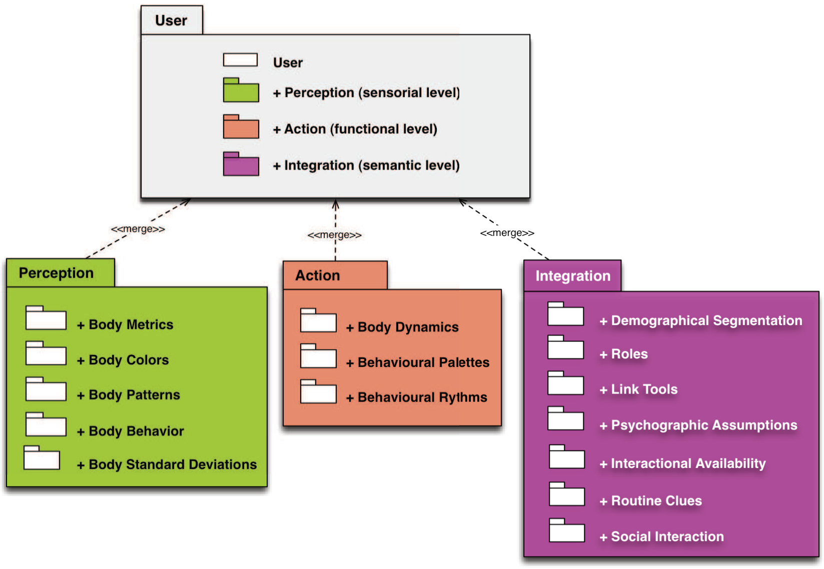


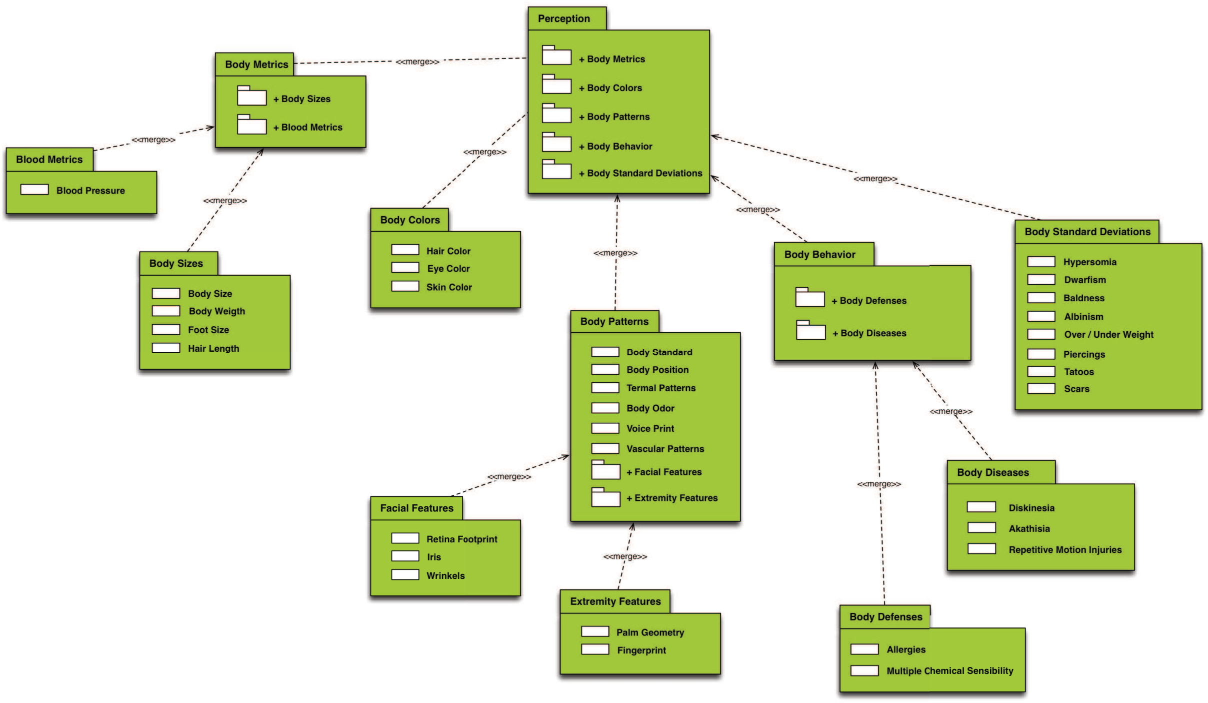

5 Blocked UI Handling
Responsible: State Manager
Responsible: Controller Component
Responsible: Data Component
Responsible: Interaction Manager
UIUpdateNotification
state = HAS_CONTEXT
data = {'state':'5'}
yes
PROCESS MUST_INFORM
no
PROCESS INPUT
yes
HAS_INPUT
PROCESS MMI REQUEST RECEIVED
no
yes
PROCESS clearContextRequest
yes
no
PROCESS pauseRequest
yes
no
PROCESS cancelRequest
no
yes
PROCESS pause
yes
no
PROCESS cancel
no
yes
SEND extensionNotification
CURRENT COMMAND cancel
CURRENT COMMAND clearContext
CURRENT COMMAND pause
DELETE context
LOGGING save UI state
BLOCKED UI cancel semantics
BLOCKED UI pause semantics
UPDATED
start
PROCESS CURRENT COMMAND
prepare
SEND StartResponse
SEND PrepareResponse
go to Context Handling
go to Update UI Handling
end Blocked UI Handling

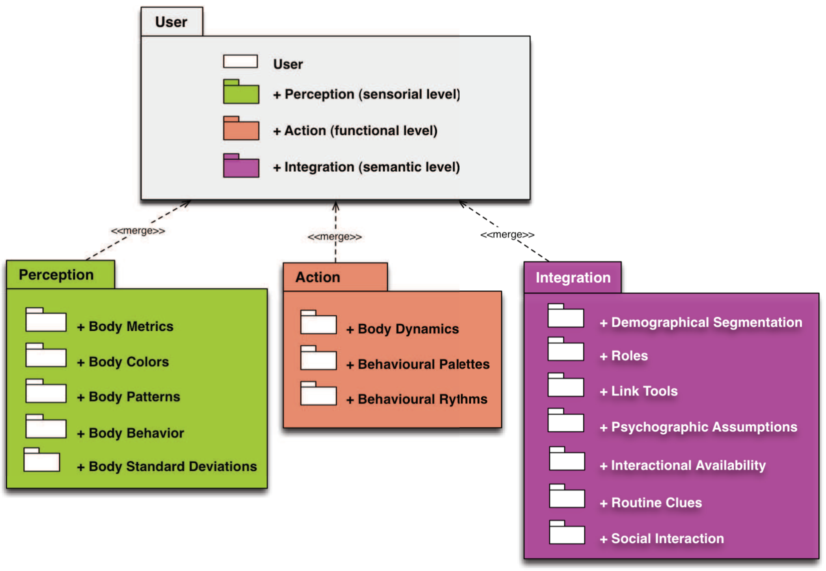

6 Updated UI Handling
Responsible: State Manager
Responsible: Controller Component
Responsible: Data Component
Responsible: Interaction Manager
UIUpdateNotification
state = HAS_CONTEXT
data = {'state':'5'}
yes
PROCESS MUST_INFORM
no
PROCESS INPUT
yes
HAS_INPUT
PROCESS MMI REQUEST RECEIVED
no
yes
PROCESS prepareRequest
yes
no
PROCESS resumeRequest
yes
no
PROCESS startRequest
no
yes
PROCESS prepare
yes
no
PROCESS resume
yes
no
PROCESS start
yes
no
PROCESS other semantics
no
yes
SEND extensionNotification
CURRENT COMMAND start
CURRENT COMMAND prepare
CURRENT COMMAND resume
LOAD data
LOAD saved state
UPDATE UI start semantics
UPDATE UI resume semantics
UPDATED
prepare
PROCESS CURRENT COMMAND
start
SEND startResponse
SEND prepareResponse
SEND resumeResponse
return to Updated UI Handling start
go to Blocked UI Handling
end Updated UI Handling

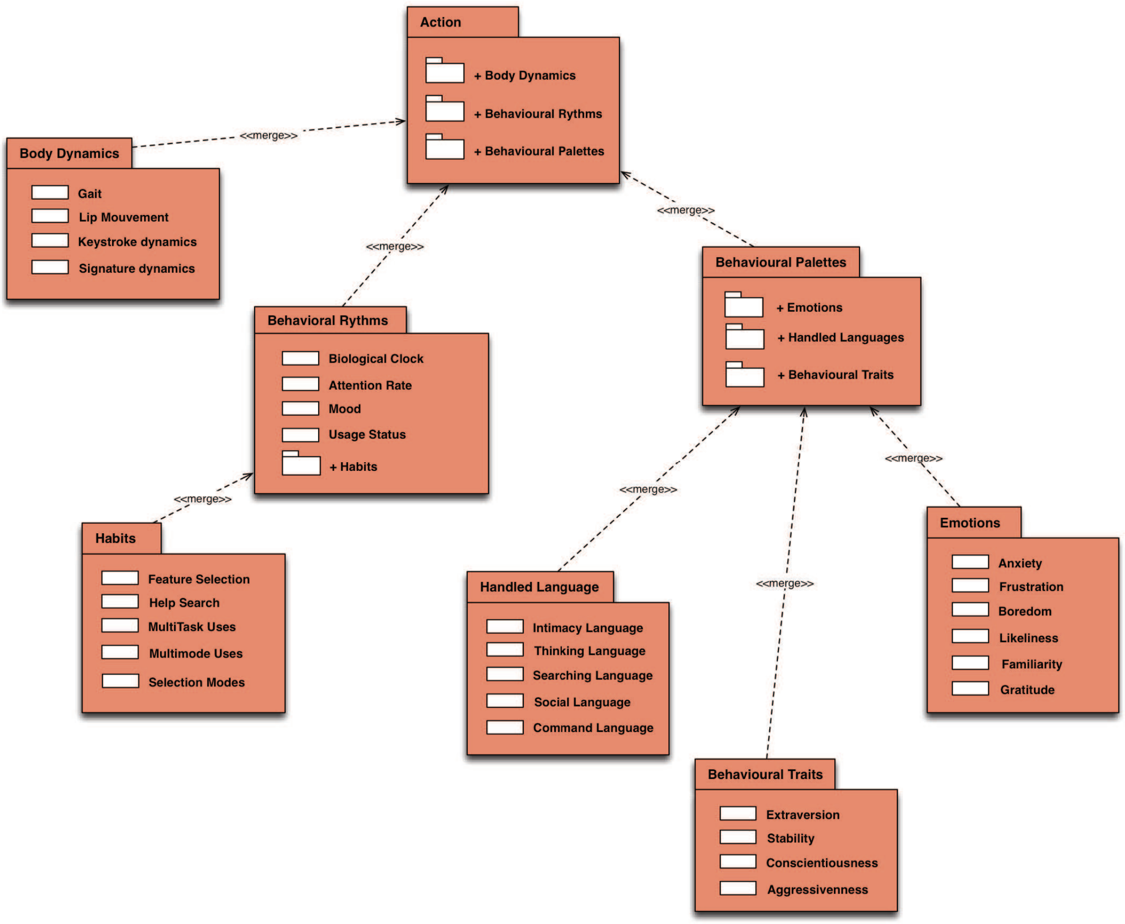
4 User Model

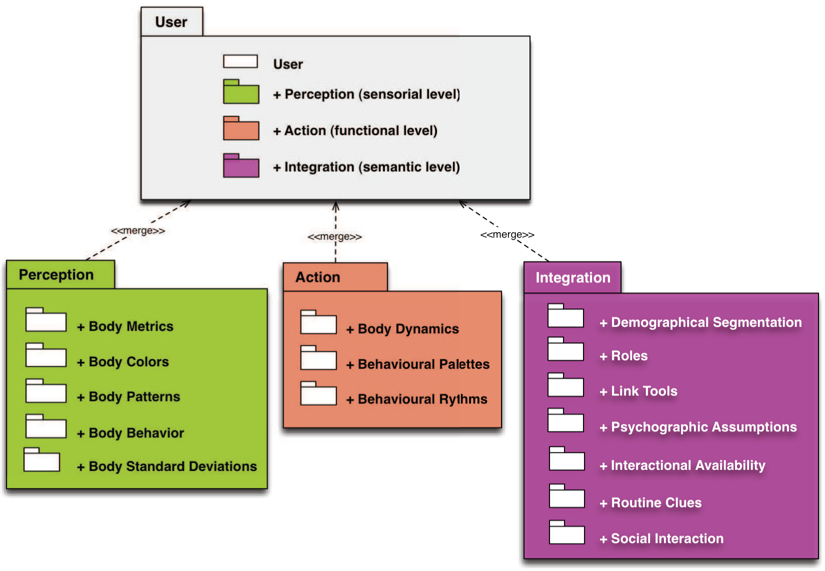
The following user model is based in the analysis of some of the standards for user modeling presented in section 1.2.2.2. User modeling and the use of the conceptual layers of the Soa2m Situation Model to structure (and collect) the information according with the granularities used in this thesis. The result is a detailed vocabulary and data model for the description of users in current multimodal systems.

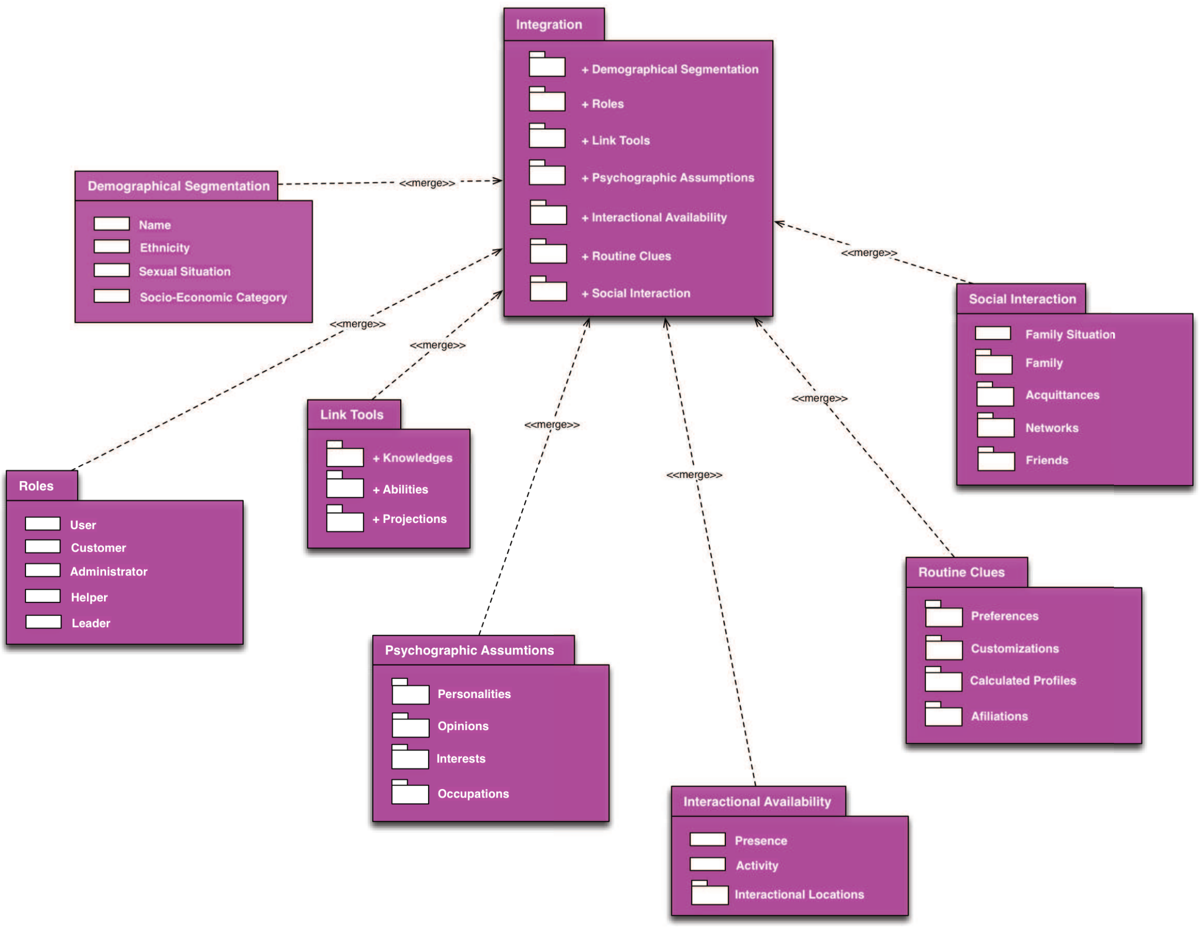





User
User
+ Perception (sensorial level)
+ Action (functional level)
+ Integration (semantic level)
<<merge>>
<<merge>>
<<merge>>
Perception
+ Body Metrics
+ Body Colors
+ Body Patterns
+ Body Behavior
+ Body Standard Deviations
Action
+ Body Dynamics
+ Behavioural Palettes
+ Behavioural Rythms
Integration
+ Demographical Segmentation
+ Roles
+ Link Tools
+ Psychographic Assumptions
+ Interactional Availability
+ Routine Clues
+ Social Interaction


Action
+ Body Dynamics
+ Behavioural Rythms
+ Behavioural Palettes
Body Dynamics
Gait
Lip Mouvement
Keystroke dynamics
Signature dynamics
<<merge>>
<<merge>>
<<merge>>
Behavioral Rythms
Biological Clock
Attention Rate
Mood
Usage Status
+ Habits
Behavioural Palettes
+ Emotions
+ Handled Languages
+ Behavioural Traits
<<merge>>
Habits
Feature Selection
Help Search
MultiTask Uses
Multimode Uses
Selection Modes
<<merge>>
<<merge>>
<<merge>>
Handled Language
Intimacy Language
Thinking Language
Searching Language
Social Language
Command Language
Emotions
Anxiety
Frustration
Boredom
Likeliness
Familiarity
Gratitude
Behavioural Traits
Extraversion
Stability
Conscientiousness
Aggressivenness


User
User
+ Perception (sensorial level)
+ Action (functional level)
+ Integration (semantic level)
<<merge>>
<<merge>>
<<merge>>
Perception
+ Body Metrics
+ Body Colors
+ Body Patterns
+ Body Behavior
+ Body Standard Deviations
Action
+ Body Dynamics
+ Behavioural Palettes
+ Behavioural Rythms
Integration
+ Demographical Segmentation
+ Roles
+ Link Tools
+ Psychographic Assumptions
+ Interactional Availability
+ Routine Clues
+ Social Interaction


Integration
+ Demographical Segmentation
+ Roles
+ Link Tools
+ Psychographic Assumptions
+ Interactional Availability
+ Routine Clues
+ Social Interaction
Demographical Segmentation
Name
Ethnicity
Sexual Situation
Socio-Economic Category
<<merge>>
<<merge>>
<<merge>>
<<merge>>
<<merge>>
<<merge>>
<<merge>>
Social Interaction
Family Situation
Family
Acquittances
Networks
Friends
Roles
User
Customer
Administrator
Helper
Leader
Link Tools
+ Knowledges
+ Abilities
+ Projections
Psychographic Assumtions
Personalities
Opinions
Interests
Occupations
Interactional Availability
Presence
Activity
Interactional Locations
Routine Clues
Preferences
Customizations
Calculated Profiles
Affiliations

5 The Soa2m Situation

The following code is the ontology file to be imported as an ontology design pattern to annotate situations:

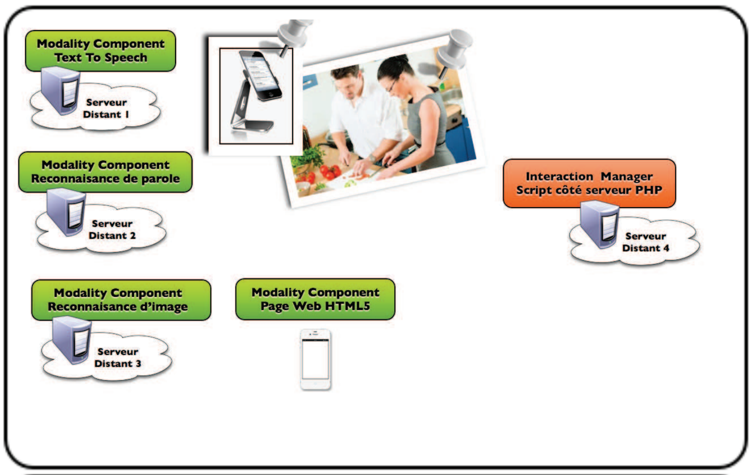
```
<rdf:RDF xmlns="http://localhost:8888/soa2m/ont/srcs/SOA2M/situation/Soa2mSituation.owl#"
    xml:base="http://localhost:8888/soa2m/ont/srcs/SOA2M/situation/Soa2mSituation.owl"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:DOLCE-Lite="http://localhost:8888/soa2m/ont/srcs/DOLCE/DLP/DOLCE-Lite.owl#"
    xmlns:how="http://localhost:8888/soa2m/ont/srcs/SOA2M/description/how.owl#"
    xmlns:Soa2mSituation="http://localhost:8888/soa2m/ont/srcs/SOA2M/situation/Soa2mSituation.owl#"
    xmlns:TemporalRelations="http://localhost:8888/soa2m/ont/srcs/DOLCE/DLP/TemporalRelations.owl#"
    xmlns:who="http://localhost:8888/soa2m/ont/srcs/SOA2M/description/who.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:which="http://localhost:8888/soa2m/ont/srcs/SOA2M/description/which.owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:SpatialRelations="http://localhost:8888/soa2m/ont/srcs/DOLCE/DLP/SpatialRelations.owl#"
    xmlns:ExtendedDnS="http://localhost:8888/soa2m/ont/srcs/DOLCE/DLP/ExtendedDnS.owl#">
   <owl:Ontology rdf:about="http://localhost:8888/soa2m/ont/srcs/SOA2M/situation/Soa2mSituation.owl">
      <rdfs:label rdf:datatype="&xsd;string">The Multimodal Situation Description</rdfs:label>
      <owl:versionInfo rdf:datatype="&xsd;string">1.0</owl:versionInfo>
      <owl:versionInfo>Version 1.0.: Basic Inheritance. </owl:versionInfo>
      <dc:rights>@copyright Copyright (c) 2012 B.Helena RODRIGUEZ - soixante-dix.com l Telecom ParisTech.</dc:rights>
      <dc:creator>Created by  B.Helena RODRIGUEZ, based on DOLCE-Lite 3.9.7</dc:creator>
      <owl:imports rdf:resource="http://localhost:8888/soa2m/ont/srcs/DOLCE/DLP/DOLCE-Lite.owl"/>
      <owl:imports rdf:resource="http://localhost:8888/soa2m/ont/srcs/DOLCE/DLP/ExtendedDnS.owl"/>
      <owl:imports rdf:resource="http://localhost:8888/soa2m/ont/srcs/SOA2M/description/how.owl"/>
      <owl:imports rdf:resource="http://localhost:8888/soa2m/ont/srcs/SOA2M/description/which.owl"/>
      <owl:imports rdf:resource="http://localhost:8888/soa2m/ont/srcs/SOA2M/description/who.owl"/>
   </owl:Ontology>

    <owl:AnnotationProperty rdf:about="&dc;creator"/>
   <owl:AnnotationProperty rdf:about="&owl;versionInfo"/>
   <owl:AnnotationProperty rdf:about="&rdfs;label"/>
   <owl:AnnotationProperty rdf:about="&dc;rights"/>

   <!--  / Object Properties   -->

  <owl:ObjectProperty rdf:about="&DOLCE-Lite;generic-constituent-of"/>

   <owl:ObjectProperty rdf:about="&DOLCE-Lite;mediated-relation"/>

   <owl:ObjectProperty rdf:about="&DOLCE-Lite;part"/>

   <owl:ObjectProperty rdf:about="&DOLCE-Lite;part-of"/>

   <owl:ObjectProperty rdf:about="&DOLCE-Lite;participant"/>

   <owl:ObjectProperty rdf:about="&DOLCE-Lite;participant-in"/>

   <owl:ObjectProperty rdf:about="&DOLCE-Lite;specific-constant-constituent"/>



   <owl:ObjectProperty rdf:about="&ExtendedDnS;references"/>

   <owl:ObjectProperty rdf:about="&ExtendedDnS;requires"/>

   <owl:ObjectProperty rdf:about="&ExtendedDnS;specialized-by"/>

   <owl:ObjectProperty rdf:about="&ExtendedDnS;specializes"/>

   <owl:ObjectProperty rdf:about="&SpatialRelations;d-spatial-location"/>

   <owl:ObjectProperty rdf:about="&TemporalRelations;present-at"/>

   <owl:ObjectProperty rdf:about="&Soa2mSituation;hasDescription">
      <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;requires"/>
      <rdfs:range rdf:resource="&Soa2mSituation;Soa2mDescription"/>
      <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mSituation"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="&Soa2mSituation;hasFacet">
      <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;requires"/>
      <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mSituation"/>
      <rdfs:range>
         <owl:Class>
            <owl:unionOf rdf:parseType="Collection">
               <rdf:Description rdf:about="&Soa2mSituation;behavioralFacet"/>
               <rdf:Description rdf:about="&Soa2mSituation;intentionalFacet"/>
               <rdf:Description rdf:about="&Soa2mSituation;semanticFacet"/>
               <rdf:Description rdf:about="&Soa2mSituation;sensorialFacet"/>
            </owl:unionOf>
         </owl:Class>
      </rdfs:range>
   </owl:ObjectProperty>
```

```

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasGranularity">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;requires"/>
   <rdfs:range rdf:resource="&Soa2mSituation;Soa2mGranularity"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mSituation"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasHowExecution">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;requires"/>
   <rdfs:range rdf:resource="&how;executionValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasHowIntention">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&how;intentionValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasHowPerformance">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&how;performanceValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasWhat">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&which;whatValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about="&Soa2mSituation;hasWhen">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&which;whenValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasWhere">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&which;whereValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasWhoActor">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&who;actorValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasWhoInspirator">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&who;inspiratorValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;hasWhoOriginator">
   <rdfs:subPropertyOf rdf:resource="&ExtendedDnS;references"/>
   <rdfs:range rdf:resource="&who;originatorValue"/>
   <rdfs:domain rdf:resource="&Soa2mSituation;Soa2mDescription"/>
</owl:ObjectProperty>

<owl:ObjectProperty rdf:about="&Soa2mSituation;temporally-started">
   <rdfs:subPropertyOf rdf:resource="&DOLCE-Lite;mediated-relation-i"/>
   <rdfs:range rdf:resource="&DOLCE-Lite;particular"/>
   <rdfs:domain rdf:resource="&DOLCE-Lite;temporal-region"/>
   <owl:inverseOf rdf:resource="&Soa2mSituation;temporally-starts"/>
</owl:ObjectProperty>


<!--  Classes    -->

<owl:Class rdf:about="&DOLCE-Lite;event"/>

<owl:Class rdf:about="&DOLCE-Lite;temporal-region"/>

<owl:Class rdf:about="&ExtendedDnS;cognitive-event"/>

<owl:Class rdf:about="&ExtendedDnS;communication-event"/>
```

```

<owl:Class rdf:about="&ExtendedDnS;desire"/>

<owl:Class rdf:about="&ExtendedDnS;information-encoding-system">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;generic-constituent-of"/>
            <owl:someValuesFrom rdf:resource="&Soa2mSituation;Soa2mFacet"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&ExtendedDnS;modal-description"/>

<owl:Class rdf:about="&ExtendedDnS;practice"/>

<owl:Class rdf:about="&ExtendedDnS;situation"/>

<owl:Class rdf:about="&ExtendedDnS;social-description"/>

<owl:Class rdf:about="&how;executionValue"/>

<owl:Class rdf:about="&how;how">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mDescription"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&how;intentionValue"/>

<owl:Class rdf:about="&how;performanceValue"/>

<owl:Class rdf:about="&which;what">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;requires"/>
            <owl:someValuesFrom rdf:resource="&which;when"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&SpatialRelations;d-spatial-location"/>
            <owl:someValuesFrom rdf:resource="&which;whereValue"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;requires"/>
            <owl:someValuesFrom rdf:resource="&which;where"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&TemporalRelations;present-at"/>
            <owl:someValuesFrom rdf:resource="&which;whenValue"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&which;whatValue"/>

<owl:Class rdf:about="&which;when"/>

<owl:Class rdf:about="&which;whenValue"/>

<owl:Class rdf:about="&which;where"/>

<owl:Class rdf:about="&which;whereValue"/>

<owl:Class rdf:about="&which;which">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mDescription"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;specializes"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mDescription"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&who;actorValue"/>

<owl:Class rdf:about="&who;inspiratorValue"/>

<owl:Class rdf:about="&who;originatorValue"/>

<owl:Class rdf:about="&who;who">
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mDescription"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;Soa2mDescription">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;situation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mSituation"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;specialized-by"/>
            <owl:allValuesFrom>
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&how;how"/>
                        <rdf:Description rdf:about="&which;which"/>
                        <rdf:Description rdf:about="&who;who"/>
                    </owl:intersectionOf>
                </owl:Class>
            </owl:allValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;specific-constant-constituent"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:intersectionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&how;how"/>
                        <rdf:Description rdf:about="&which;which"/>
                        <rdf:Description rdf:about="&who;who"/>
                    </owl:intersectionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;Soa2mFacet">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;situation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;specific-constant-constituent"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&Soa2mSituation;behavioralFacet"/>
                        <rdf:Description rdf:about="&Soa2mSituation;intentionalFacet"/>
                        <rdf:Description rdf:about="&Soa2mSituation;semanticFacet"/>
                        <rdf:Description rdf:about="&Soa2mSituation;sensorialFacet"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;specialized-by"/>
            <owl:allValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&Soa2mSituation;behavioralFacet"/>
                        <rdf:Description rdf:about="&Soa2mSituation;intentionalFacet"/>
                        <rdf:Description rdf:about="&Soa2mSituation;semanticFacet"/>
                        <rdf:Description rdf:about="&Soa2mSituation;sensorialFacet"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:allValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mSituation"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;Soa2mGranularity">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;situation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mSituation"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;specialized-by"/>
            <owl:allValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&Soa2mSituation;punctualGranularity"/>
                        <rdf:Description rdf:about="&Soa2mSituation;relationalGranularity"/>
                        <rdf:Description rdf:about="&Soa2mSituation;symbolicGranularity"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:allValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;specific-constant-constituent"/>
            <owl:someValuesFrom>
                <owl:Class>
                    <owl:unionOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="&Soa2mSituation;punctualGranularity"/>
                        <rdf:Description rdf:about="&Soa2mSituation;relationalGranularity"/>
                        <rdf:Description rdf:about="&Soa2mSituation;symbolicGranularity"/>
                    </owl:unionOf>
                </owl:Class>
            </owl:someValuesFrom>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;Soa2mSituation">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;situation"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;participant-in"/>
            <owl:someValuesFrom rdf:resource="&Soa2mSituation;Soa2mSituationValue"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part"/>
            <owl:allValuesFrom>
                <owl:Class>
```

```xml
                <owl:intersectionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="&Soa2mSituation;Soa2mDescription"/>
                    <rdf:Description rdf:about="&Soa2mSituation;Soa2mFacet"/>
                    <rdf:Description rdf:about="&Soa2mSituation;Soa2mGranularity"/>
                </owl:intersectionOf>
            </owl:Class>
        </owl:allValuesFrom>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;Soa2mSituationValue">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;communication-event"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;participant"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mSituation"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;behavioralFacet">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;information-encoding-system"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;generic-constituent-of"/>
            <owl:someValuesFrom rdf:resource="&Soa2mSituation;Soa2mFacet"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;specializes"/>
            <owl:allValuesFrom rdf:resource="&ExtendedDnS;practice"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;intentionalFacet">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;information-encoding-system"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;generic-constituent-of"/>
            <owl:someValuesFrom rdf:resource="&Soa2mSituation;Soa2mFacet"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&ExtendedDnS;specializes"/>
            <owl:allValuesFrom rdf:resource="&ExtendedDnS;desire"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;punctualGranularity">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;information-encoding-system"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;generic-constituent-of"/>
            <owl:someValuesFrom rdf:resource="&Soa2mSituation;Soa2mGranularity"/>
        </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mGranularity"/>
        </owl:Restriction>
    </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="&Soa2mSituation;relationalGranularity">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;modal-description"/>
    <rdfs:subClassOf>
        <owl:Restriction>
            <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
            <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mGranularity"/>
        </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:about="&Soa2mSituation;semanticFacet">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;information-encoding-system"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&ExtendedDnS;specializes"/>
        <owl:allValuesFrom rdf:resource="&ExtendedDnS;social-description"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&DOLCE-Lite;generic-constituent-of"/>
        <owl:someValuesFrom rdf:resource="&Soa2mSituation;Soa2mFacet"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:about="&Soa2mSituation;sensorialFacet">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;information-encoding-system"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&DOLCE-Lite;generic-constituent-of"/>
        <owl:someValuesFrom rdf:resource="&Soa2mSituation;Soa2mFacet"/>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&ExtendedDnS;specializes"/>
        <owl:allValuesFrom rdf:resource="&ExtendedDnS;cognitive-event"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>

  <owl:Class rdf:about="&Soa2mSituation;symbolicGranularity">
    <rdfs:subClassOf rdf:resource="&ExtendedDnS;social-description"/>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="&DOLCE-Lite;part-of"/>
        <owl:allValuesFrom rdf:resource="&Soa2mSituation;Soa2mGranularity"/>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
</rdf:RDF>
```

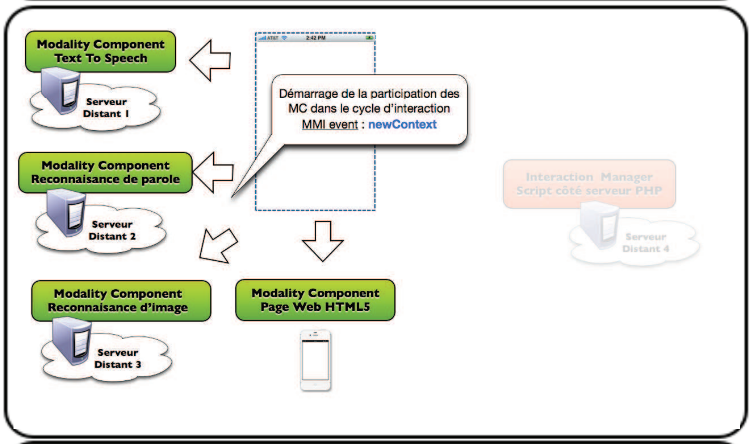
6 Soa2m Use-Case


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
Modality Component Page Web HTML5
Interaction Manager Script côté serveur PHP
Serveur Distant 4

A person is working in the kitchen. His hands are not available and he wants to watch TV on his smartphone.

The system has three available modalities implemented in distant servers and one modality displayed on the smartphone.

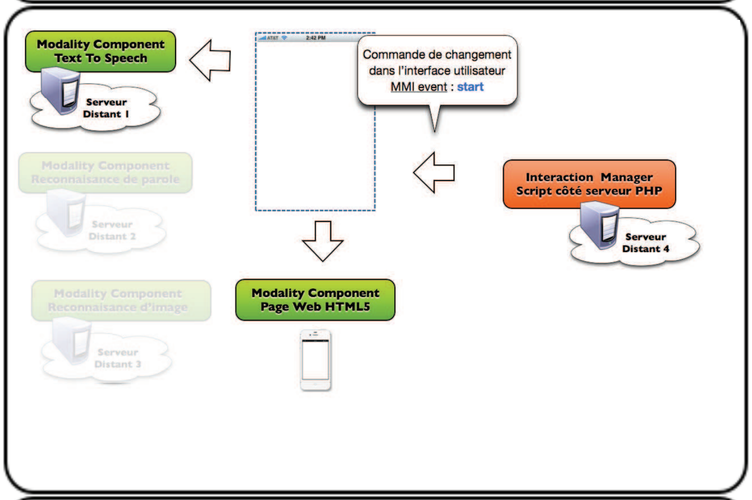
The Interaction Manager is also located on a distant server.


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
Modality Component Page Web HTML5
Démarrage de la participation des MC dans le cycle d'interaction
MMI event : newContext
Interaction Manager Script côté serveur PHP
Serveur Distant 4

STEP 1 The user waits(loadTime)

A new context of interaction is created by the Interaction manager and transmitted to the Modality Components to «connect» them and to prepare them to start the interaction.

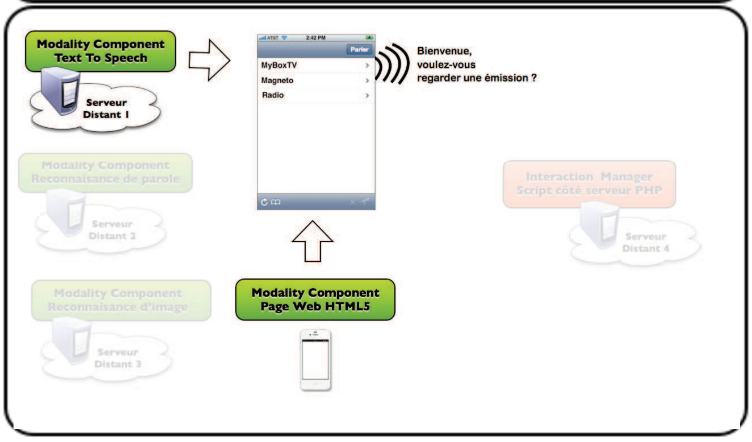
This is made using a newContext event.


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
Commande de changement dans l'interface utilisateur
MMI event : start
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Page Web HTML5

STEP 2 The user waits (loadTime)

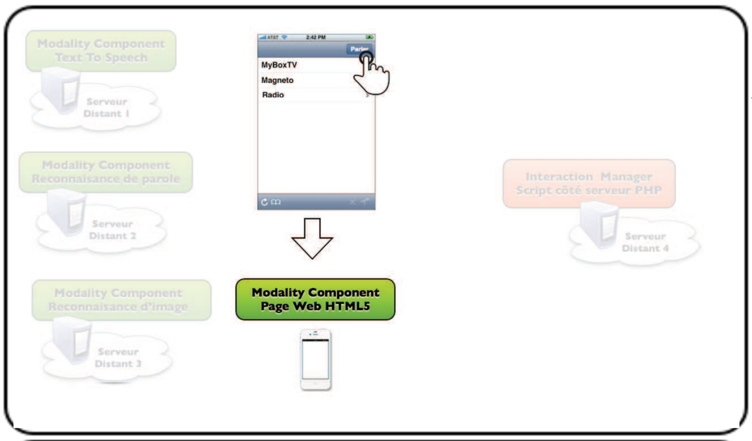
The interaction cycle starts and the command to load the first step of the user interface is sent to the two Modality Components used on this first step.

This is made using a start event sent by the Interaction Manager.


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
Parler
MyBoxTV
Magneto
Radio
Bienvenue, voulez-vous regarder une émission ?
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Page Web HTML5

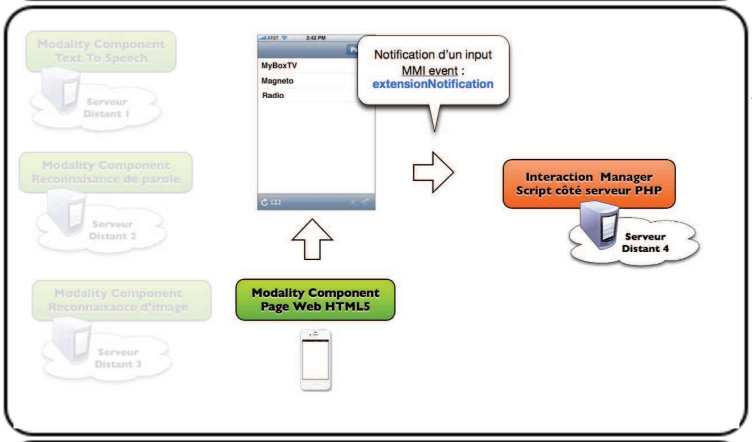
STEP 3 The applications plays a welcome audio message and asks for a voice input : «Do you want to watch a show?»

The text to Speech Modality Component uses the audio API's to reproduce the synthesized welcome sound. The Web Page Modality Component displays the first step of the GUI.


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
MyBoxTV
Magneto
Radio
Parler
Modality Component Page Web HTML5
Interaction Manager Script côté serveur PHP
Serveur Distant 4

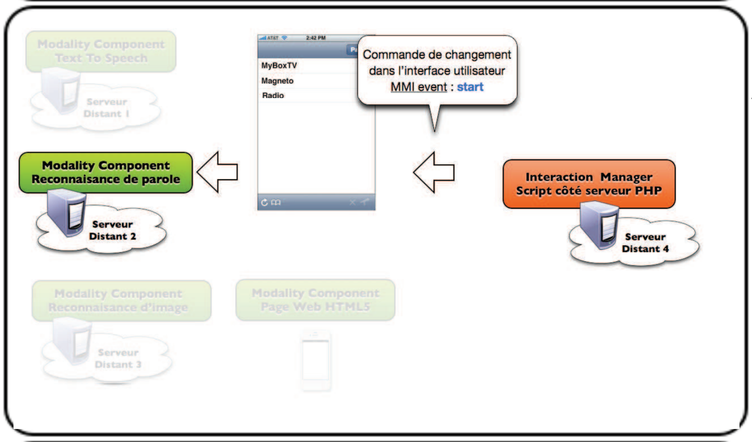
STEP 4 The user touches the button «speak»

The GUI in the Web Page Modality Component receives the touch event informing that the user will talk to the system.


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
MyBoxTV
Magneto
Radio
Notification d'un input
MMI event :
extensionNotification
Modality Component Page Web HTML5
Interaction Manager Script côté serveur PHP
Serveur Distant 4

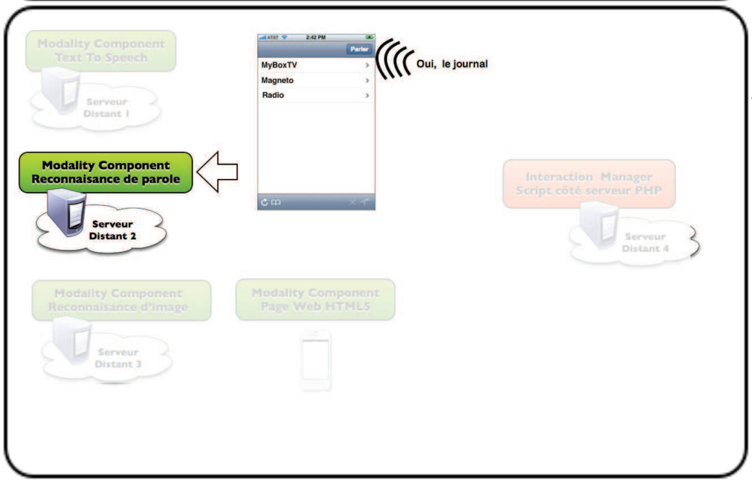
STEP 5 The user prepares itself to speak to the system

The Web Page Modality Component informs the Interaction Manager of the user input.


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
MyBoxTV
Magneto
Radio
Commande de changement dans l'interface utilisateur
MMI event : start
Modality Component Page Web HTML5
Interaction Manager Script côté serveur PHP
Serveur Distant 4

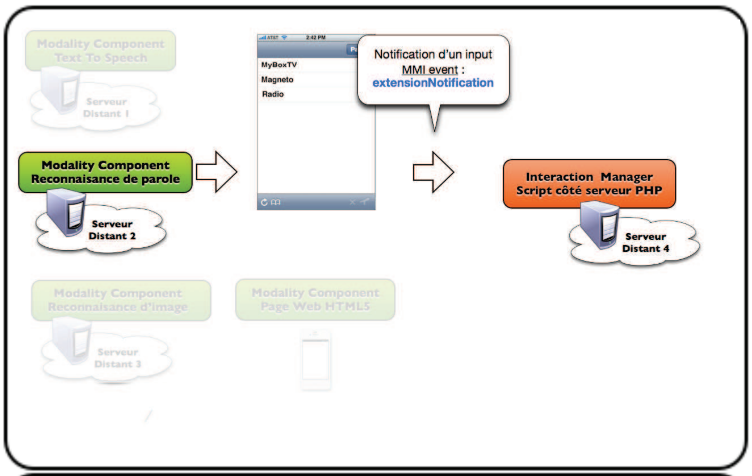
STEP 6 The user prepares itself to speak to the system

The Interaction Manager authorizes the user interface to pass to the next step. It sends a start event to the Speech Recognizer Modality Component that prepares the processes needed to recognize the user's voice.


Modality Component Text To Speech
Serveur Distant 1
Modality Component Reconnaissance de parole
Serveur Distant 2
Modality Component Reconnaissance d'image
Serveur Distant 3
MyBoxTV
Magneto
Radio
Parler
Oui, le journal
Modality Component Page Web HTML5
Interaction Manager Script côté serveur PHP
Serveur Distant 4

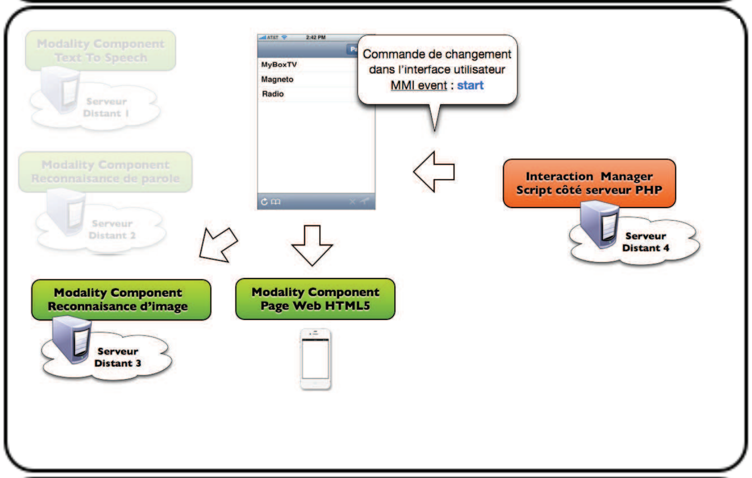
STEP 7 The user speaks to the system and responds : «yes, the news»

The Speech Recognizer Modality Component recognizes the user's utterances (*feature fusion*).


Modality Component Text To Speech
Serveur Distant 1
MyBoxTV
Magneto
Radio
Notification d'un input
MMI event : extensionNotification
Modality Component Reconnaisance de parole
Serveur Distant 2
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Reconnaissance d'image
Serveur Distant 3
Modality Component Page Web HTML5

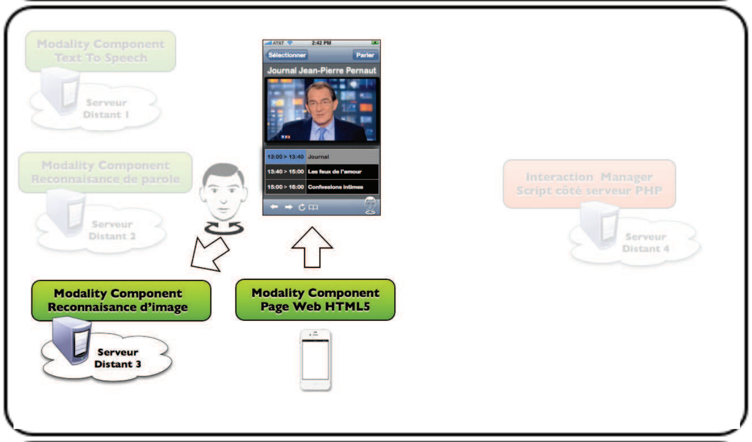
STEP 8 The user waits

The Speech Recognizer Modality Component informs the Interaction Manager of the user input, and sends the resulting data (recognized x best results).


Modality Component Text To Speech
Serveur Distant 1
MyBoxTV
Magneto
Radio
Commande de changement dans l'interface utilisateur
MMI event : start
Modality Component Reconnaisance de parole
Serveur Distant 2
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Reconnaissance d'image
Serveur Distant 3
Modality Component Page Web HTML5

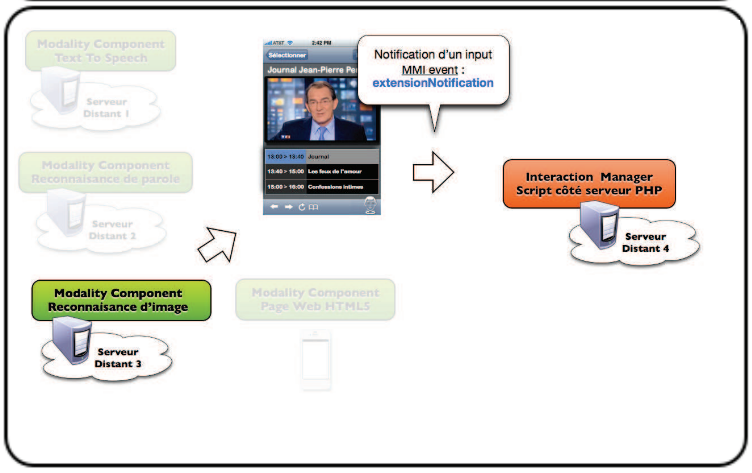
STEP 9 The user waits

The Interaction Manager recognizes the command and authorizes the user interface to pass to the next step. It sends a start event to the Image Recognizer Modality Component that prepares the processes needed to recognize the user's face. It also sends a start event to the GUI in the web page to update the display to the next phase of the GUI corresponding to the interpreted command.


Modality Component Text To Speech
Serveur Distant 1
Sélectionner
Parler
Journal Jean-Pierre Pernaut
13:00 > 13:40 Journal
13:40 > 15:00 Les feux de l'amour
15:00 > 16:00 Confessions intimes
Modality Component Reconnaisance de parole
Serveur Distant 2
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Reconnaissance d'image
Serveur Distant 3
Modality Component Page Web HTML5

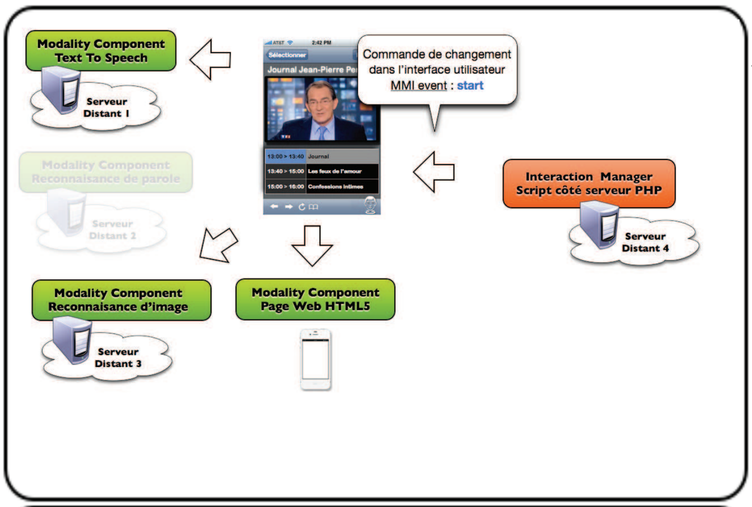
STEP 10 The user watches the news and decides to channel surf. He turns his face in front of the phone cam.

The Image Recognizer Modality Component recognizes the user's movement and matches the movement with the semantics of the command(*semantic fusion*).


Modality Component Text To Speech
Serveur Distant 1
Sélectionner
Journal Jean-Pierre Pe
13:00 > 13:40 Journal
13:40 > 15:00 Les feux de l'amour
15:00 > 16:00 Confessions intimes
Notification d'un input
MMI event : extensionNotification
Modality Component Reconnaisance de parole
Serveur Distant 2
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Reconnaissance d'image
Serveur Distant 3
Modality Component Page Web HTML5

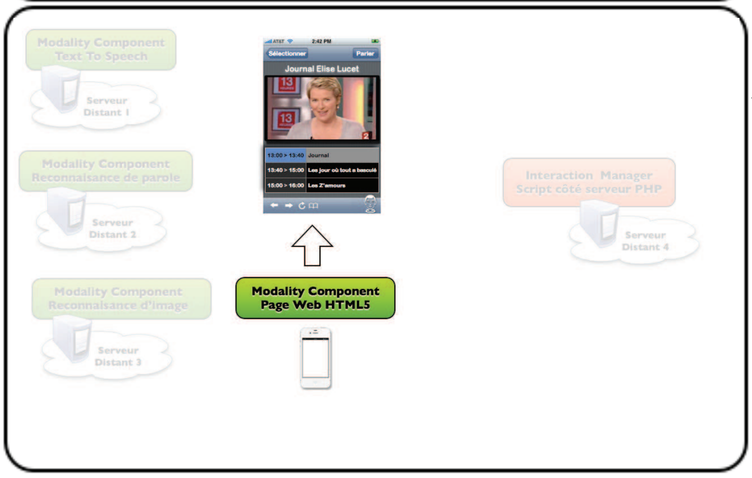
STEP 11 The user waits

The Image Recognizer Modality Component informs the Interaction Manager of the user input, and sends the recognized result data.


Modality Component Text To Speech
Serveur Distant 1
Commande de changement dans l'interface utilisateur MMI event : start
Modality Component Reconnaisance de parole
Serveur Distant 2
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Reconnaisance d'image
Serveur Distant 3
Modality Component Page Web HTML5

STEP 12 The user waits

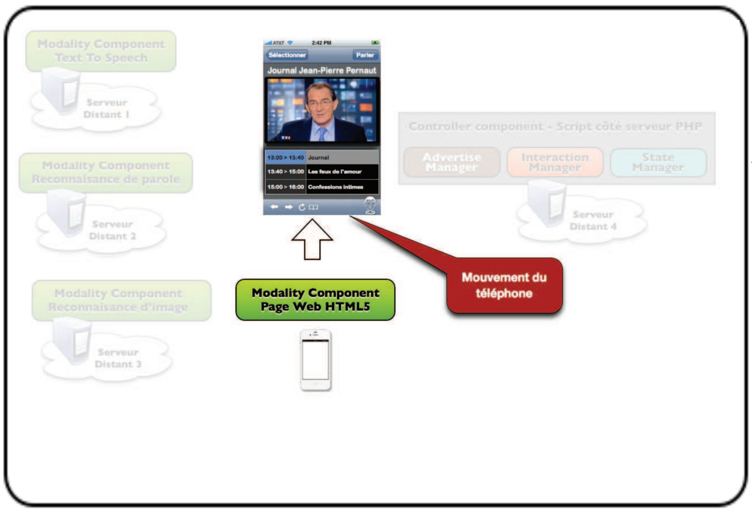
The Interaction Manager recognizes the command and authorizes the user interface to pass to the next step. It sends a start event to the Image Recognizer Modality Component to allow it to continue its work. It also sends a start event to the GUI in the web page to update the display to the next phase of the GUI corresponding to the interpreted zapping command. It also prevents the Text to Speech Modality Component that it can be requested.


Modality Component Text To Speech
Serveur Distant 1
Journal Elise Lucet
Modality Component Reconnaisance de parole
Serveur Distant 2
Interaction Manager Script côté serveur PHP
Serveur Distant 4
Modality Component Reconnaisance d'image
Serveur Distant 3
Modality Component Page Web HTML5

STEP 13 The user watches the next news TV show

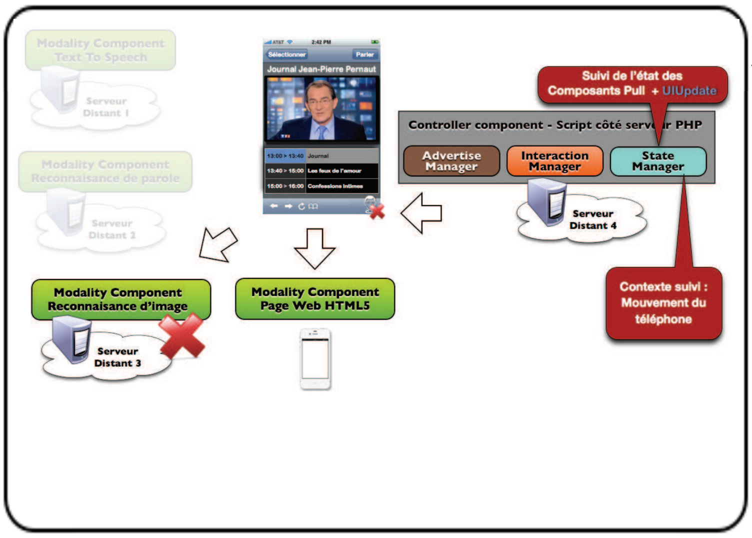
The GUI in the Web Page Modality Component receives the zapping command and updates the user interface with the new TV channel.

The following cases reflect the Soa2m contribution regarding the dynamic behavior of the system, the use of two new events to handle the state of the surrounding environment and the availability state of the Modality Components. It shows the use of the new components proposed on this Thesis to extend the MMI Framework & Architecture Recommendation.


Modality Component Text To Speech
Serveur Distant 1
Journal Jean-Pierre Pernaut
Controller component - Script côté serveur PHP
Advertise Manager
Interaction Manager
State Manager
Modality Component Reconnaisance de parole
Serveur Distant 2
Serveur Distant 4
Mouvement du téléphone
Modality Component Reconnaisance d'image
Serveur Distant 3
Modality Component Page Web HTML5

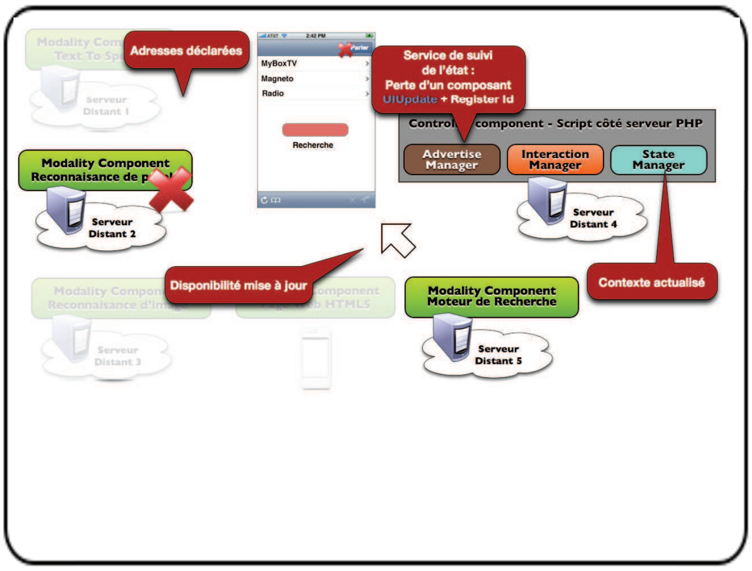
STEP 10-Bis1 The user watches the news. Then he takes the smartphone from its support and goes to another room.

The GUI in the Web Page receives an event of the accelerometer. It sends a UIUpdate notification to the Advertise Manager in the Controller Component to inform of this change on its environment state.



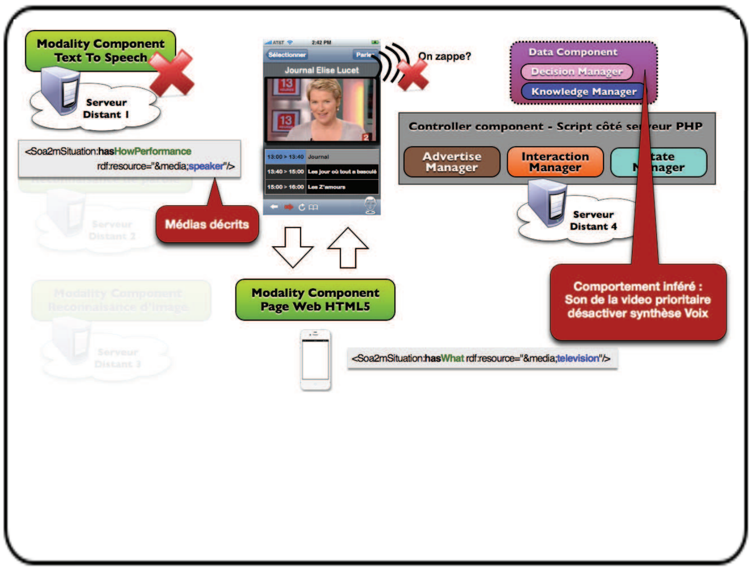
STEP 10-Bis2 The user walks

The State Manager updates the state information. The Controller Component requests a service provided by the Data Component with this new information and the Decision Manager decides form the usage description of the Image Recognizer that it must be stopped and «disconnected» from the interaction cycle. Then the Controller Component sends a UIUpdate to the GUI Modality Component to hide the icon related to the image recognizer.



STEP 3-Bis1 The user will respond to the welcome sound

The Speech Recognizer is no more available. The Advertise Manager Service does not receive UIUpdates. The State Manager updates the state information. The Controller Component checks if another modality component of the same type is available and informs this change to the Interaction Manager. Then the Controller Component updates the GUI with the new Search Modality Component data. It also hides the icons related to the Speech Recognizer MC.



STEP 14 The user is watching the news and the channel go the the advertisement adds.

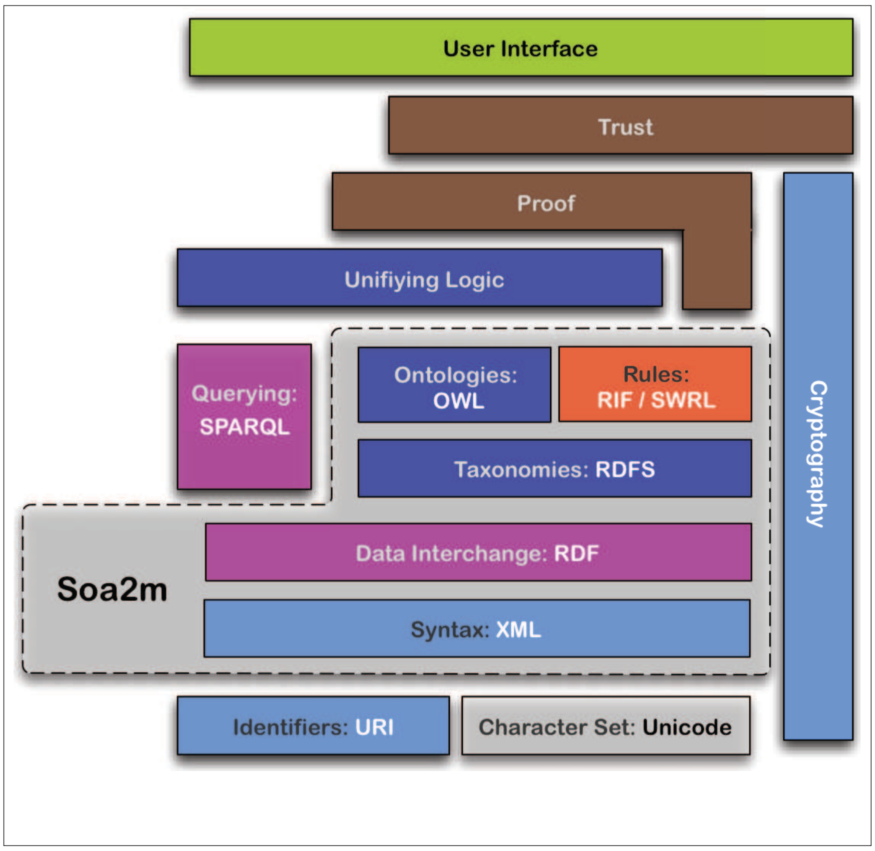
The Controller Component commands a new step on the user interface. While preparing the data the decision manager infers from the Soa2m Situation description of the Modality Components used in the interface that both use the audio system. It decides that the GUI has priority on the sound reproduction and disable the Text to Speech Modality Component. Then it updates the GUI button providing the zapping feature.

7 Semantic Web in Soa2m

- RDF / RDF-S

In the figure below, representing the official web semantic layers, the lower layer used in the Soa2m annotation method is the XML layer, that has the name-space and schema definitions that integrate the Semantic Web syntax with the other XML based standards.

The Resource Description Framework (RDF) layer represent the core data representation format for the semantic web. This is the layer allowing to make statements about objects with URI's and to define data types of the resources.



RDF is an assertional language intended to be used to express propositions and representing information about resources. It was primarily intended for representing metadata about document resources, such as the title, author, and modification date of a Web page, but it can be used for storing any other data. It is based on triples *subject-predicate-object* that form a graph of data.

The Resource Description Framework Schema (RDF-S) layer offers tools for defining knowledge models that are close to frame-based approaches. [W3C-RDFS 2004] It introduces classes, relationships of inclusion - subsumption- on both classes and properties (similar to the type systems of object-oriented programming languages); and global **domain**[186] and **range**[187] restrictions for these properties.

For example, we can define the hasHow **property** to have an Input Modality **domain** and a Touch **range** whereas a classical object-oriented programming system might typically define a class Input Modality with an attribute called how of type Touch. Using the RDF approach, it is easy for others to subsequently define additional properties with a domain of Input Modality or a range of Touch without the need to re-define the original description of these classes.

Thus, the RDF-S layer is based on a simple modeling language on top of the RDF formalism and define vocabularies of terms used for a more detailed description than RDF structures.

186 A domain restriction is an instance of a Property that is used to state that ***any resource*** having a given property is an instance of one or more classes.

187 A range restriction is an instance of a Property that is used to state that ***the values*** of a property are instances of one or more classes.

The resulting formalism allows to model ontologies as a taxonomic structure of concepts with attributes and relations to other concepts defined as properties attached to each concept. This layer can be used to create lightweight ontologies.

The Ontology layer in the figure above supports the evolution of vocabularies because it can define relations between the different concepts with the Web Ontology Language (OWL) [W3C-OWL 2009] which extends RDF and RDF-S. Its primary aim is to bring the expressive and reasoning power of Description Logic to the semantic web.

OWL comes in three levels of complexity - OWL Lite for taxonomies and simple constrains, OWL DL for a full support of description logic, and OWL Full for maximum expressiveness and syntactic freedom with the risk of incompatibility with the inference engines.

RDF-S and OWL have semantics defined that can be used for reasoning within ontologies. To provide rules beyond the constructs available from these languages, the semantic web proposes the Rules layer (see the figure above) in which rule languages are being standardized.

Two standards are emerging - RIF and SWRL. The Rule Interchange Format (RIF) is a standard for exchanging rules among rule systems, in particular among Web rule engines. RIF focused on exchange rather than trying to develop a single one-fits-all rule language.

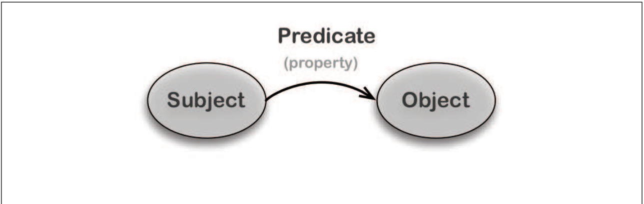
Finally, on the other hand, the Semantic Web Rule Language (SWRL) is the union[188] of an undecidable superset of OWL DL -a very expressive description logic- and a simple rules language RuleML Datalog.

SWRL includes a high-level abstract syntax for Horn-like rules[189] and all rules are expressed in terms of OWL concepts (classes, properties, individuals).

RDF uses the following key concepts: a graph data model, URI-based vocabulary, datatypes, literals[29], the XML serialization syntax, the expression of simple facts[30] and the entailment[31].

The underlying structure of any expression in RDF is a collection of triples, each consisting of a subject, a predicate (also called a property) and an object. A set of such triples is called an RDF graph.

This can be illustrated by a node and directed-arc diagram, in which each triple is represented as a node-arc-node link (a graph, see figure below). The direction of the arc is significant: it always points toward the object.



The RDF graph

Each triple represents a statement of a relationship between the things denoted by the nodes that it links. The nodes of an RDF graph are its subjects and objects.The assertion of an RDF triple says that some relationship, indicated by the predicate, holds between the things denoted by subject and object of the triple.

Thus an RDF file is a markup file that can describe some facts about the underlying resources, and linking it to a web of shared knowledge.

For example we can express that the resource, a MC Service, has as a global unique name which is the URI:

http://localhost/soa2m/desc/rdf/MC_1234#FlexVoiceSynthesizer

This resource has a property Affiliation, which is also a resource with an unique URI :

http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/architecture/MCs.rdfs#**isAffiliatedWith**

188 While Description logic, can't offer a model for the proposition «a resident student is a student who lives in the same city where he studies», Horn logic (which can model the previous statement) can't model «a person is either a man or a woman, and no person is at the same time a man and a woman» (a statement that can be modeled in description logic).

189 A Horn rule is a clause containing at most one positive literal. A rule contains an antecedent part, which is referred to as the body, and a consequent part, which is referred to as the head. A rule the the implication from the antecedent (a set of atomic clauses) to a consequent (a single atomic clause): antecedent → consequent. All the variables in the consequent must occur (be true) in at least one atom of the antecedent, and are all considered to be universally quantified (true for everything, or for every relevant thing). A SWRL rule is, for example, a property value assignment saying that if a person **p** has a sibling **s** and **s** is a man then we can infer that the person **p** has a brother: Person(?p) ^ hasSibling(?p,?s) ^ Man(?s) ⟶ hasBrother(?p,?s)

Finally, the property has a value, that can be a literal or a resource. In our example, the value is a resource:

http://localhost/soa2m/ont/participants/domain/modalities/functional/CognitiveAcoustic#Transducer

If we declare some namespaces to rewrite this URIs in a shorter form:

xmlns: **my** = http://localhost/soa2m/desc/rdf/MCs_1234

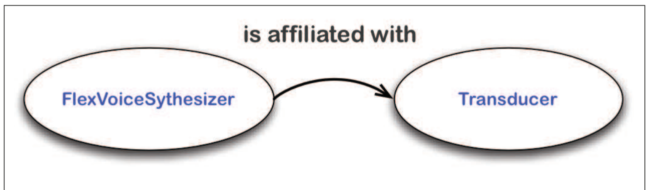
xmlns: **withProperty** = http://localhost/soa2m/ont/participants/domain/application/semantic/discrete/architecture/MCs.rdfs

xmlns: **isA** = http://localhost/soa2m/ont/participants/domain/modalities/functional/CognitiveAcoustic

Then the final RDF statement becomes:

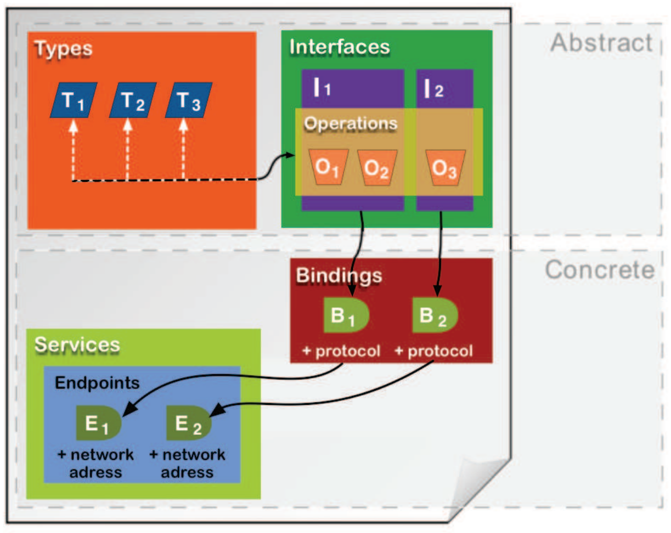
:my#FlexVoiceSynthesizer :withProperty#isAffiliatedWith :isA#Transducer

This statement corresponds to the graph in figure below and expresses that a Modality Component Service implemented with the flex media technology, and responsible of the synthesizing of a voice is related with the family of Modality Components called Transducers[190].



The RDF Affiliation Statement

- WSDL / OWL-S



The Web Services Description Language 2.0.

To describe its service, a provider can use OWL-S or he can use a WSDL file enriched with an OWL-S ontology. The WSDL provides concrete details about the service. This is called «grounding»[191] or «binding»[192] the service: to describe **how to access** the service processes.

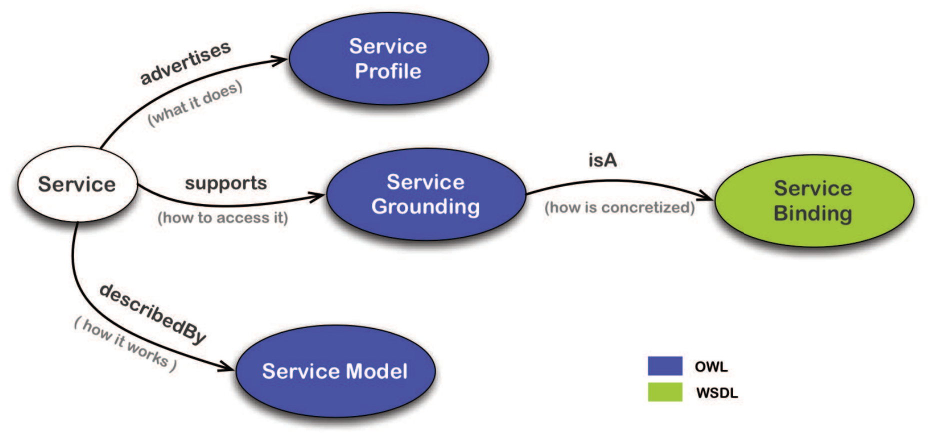
In addition to the «grounding» mechanism of WSDL, the OWL-S recommendation provides a «profiling» mechanism, which tells **what the service does**, a «process model», which tells **how the service works**.

190 A transducer is a device that converts one mode of energy to another. It can be electrical, mechanical, electromagnetic (including light), chemical, acoustic or thermal energy.

191 In OWL-S terms.

192 In WSDL terms.

The use of OWL-S and WSDL grounding involves a complementary use of the two languages. Both languages are required for a full and more expressive specification of the service, because the two languages do not cover the same conceptual space.



Complementarity between OWL and WSDL

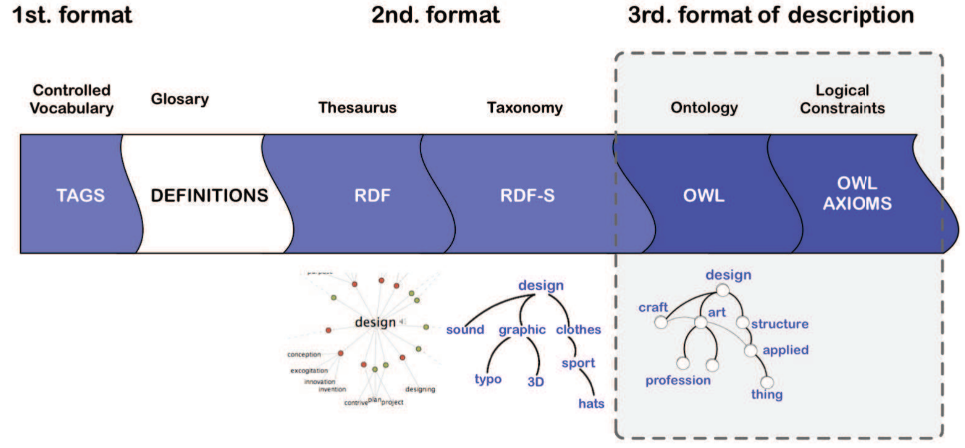
The two languages overlap in the area of the abstract layer which is used to characterize the inputs and outputs of services. WSDL specifies the abstract types using the XML Schema [W3C-XML-S 2001], whereas OWL-S allows the definition of abstract types as OWL classes based on Description Logic.

However, XSD is unable to express the semantics of an OWL class while OWL-S has no means, as currently defined, to express the binding information that WSDL captures. In result, the combined use of both standards increases the expressivity on the description of services, which is one of the Soa2m requirements to express generic services in a context-aware manner.

OWL-S is based in the OWL language [W3C-OWLS 2004] and completes the RDF and RDF-S description languages adding a formal hierarchy to the information and the construction of ontologies [**Figure 2.2.16**].

In section 2.2.2.2 Description of MC Services with RDF Manifests, we show that RDF defines a data model with sentence affirmations (triples). RDF needs a vocabulary for the data expressed in the sentences, which is a mechanisms for describing the relationships between these object, subject and predicates and other resources.

This is the role of the RDF Schema: to define classes, properties and rules that can be used to describe RDF objects, subjects and predicates. In Object Oriented Programming terminology, RDF defines instances while RDF-S defines objects and classes. In result, the RDF data model is organized in a typed hierarchy also called a taxonomy [**Figure 2.2.16**] which is a labeled graph



From Tags to Axioms in Soa2m

Nevertheless, RDF and RDF-S does not allow to describe the type of relationships expressed with the predicates (the properties) nor the axioms restricting the model.

For example, for the two RDF statements:

```
:VoiceSynthesizer   :isA   :Transducer

:Transducer         :isA   :MechanicalDevice
```

we can not express things that a human can infer with no problem, like if the properties expressed in the sentence are asymmetric (is a Transducer a VoiceSynthesizer?) or transitive (is a VoiceSynthesizer also a Mechanical Device?) neither axioms of existential quantification[193] expressed in OWL with the restriction *someValuesFrom* [**Axiom 2.2.1**] and expressing that not all the Transducers x are Mechanical Devices y.

$$\exists y \ (\ P \, (x \, , y) \wedge C \, (y) \,)$$

Axiom 2.2.1: Existential Quantification.

In the same way, using only RDF and RDFS we can not express other knowledge constraints that are common sense for humans reading the two sentences above, like the universal quantifiers [**Axiom 2.2.2**] expressed in OWL with the restriction *allValuesFrom* and expressing that all the VoiceSynthesizers x are Transducers y[194].

$$\forall y \ (\ P \, (x \, , y) \rightarrow C \, (y) \,)$$

Axiom 2.2.2: Universal Quantification.

Then, the goal of the OWL language is to overcome these limitations, representing with ontologies these types of relationships between the entities and their axioms. In Soa2m, this advantage provided by OWL will allow us to express the relationship between the different *modes* and *modalities*, for example, expressing that not all the MC services that are in acoustic mode are also continuous, or that all the MC services Listeners are also announcement Listeners, which is an equivalence relationship that we can exploit.

- SWRL

SWRL expressions can be used in the OWL-S service preconditions, and in effect expressions. Using SWRL makes the OWL-S ontologies more powerful since it uses the expressivity of rules with more extended reasoning options.

The proposed rules are of the form of an implication between an antecedent (body) and consequent (head). The rule meaning is: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold.

Both the antecedent (body) and consequent (head) consist of zero or more atoms. Atoms in these rules can be of the form C(x), P(x,y), sameAs(x,y) or differentFrom(x,y), where C is an OWL description, P is an OWL property, and x, y are either variables, OWL individuals or OWL data values.

An example of SWRL rule from the Soa2m ontologies used to represent the OWL precondition for the Start operation is:

193 ∃y means an individual y exists. C is a concept and P is a property. The axiom applied to our example announces: there exists an individual y that isA VoiceSyntesizer and also exists as a conceptual Transducer (is part of the class of Transducers). This implies that at least one individual of the Transducer concept extension (the set of all the transducers) is also a VoiceSynthesizer.

194 ∀y means for all individuals. The axiom applied to our example announces: for all the y individuals, if the property a VoiceSynthesizer isA Transducer is true then the individual is part of the Transducer concept extension (the set of all the transducers).

```
161     <process:hasPrecondition>
162         <expr:SWRL-Condition  rdf:ID="canStart">
163             <rdfs:label> hasCode( target, MC_code ) &amp; hasIdleState( target, idleState ) </rdfs:label>
164             <expr:expressionObject>
165                 <swrl:AtomList><rdf:first>
166                     <swrl:IndividualPropertyAtom>
167                         <swrl:propertyPredicate rdf:resource="&this;#hasCode"/>
168                         <swrl:argument1 rdf:resource="#target"/>
169                         <swrl:argument2 rdf:resource="#MC_code"/>
170                     </swrl:IndividualPropertyAtom> </rdf:first>
171                 <rdf:rest>
172                     <swrl:AtomList> <rdf:first>
173                         <swrl:ClassAtom>
174                             <swrl:classPredicate rdf:resource="&this;#hasIdleState"/>
175                             <swrl:argument1 rdf:resource="#target"/>
176                             <swrl:argument2 rdf:resource="#idleState"/>
177                         </swrl:ClassAtom> </rdf:first>
178                     <rdf:rest rdf:resource="&rdf;#nil"/>
179                     </swrl:AtomList>
180                 </rdf:rest>
181                 </swrl:AtomList>
182             </expr:expressionObject>
183         </expr:SWRL-Condition>
184     </process:hasPrecondition>
```

In the example we want to declare that a MC Service must launch the Start operation (which means to initiate an interaction cycle) only if the Modality Component is registered and if it is in the standBy mode (**hasIdleState**).

With this goal, the *IndividualPropertyAtom* class of SWRL is used (line 166 of the snippet code in the previous page). It consists of a *propertyPredicate* (line 167 of the snippet code in the previous page) and two arguments.

The precondition in the example has the name **canStart** and checks whether a target (a Modality Component) has a code and is in idleState. For the first Atom, arguments are of class **target** and **MC_code** and the predicate is identified with the **hasCode** property predicate (line 167 to 169 in the previous page).

For the second Atom, arguments are of class **target** and **idleState** and the predicate is identified with the **hasIdleState** property predicate (line 174 to 176 in the previous page).

The definitions of the arguments and the property predicates are imported in the OWL-S document by the linking process offered by the semantic web.

On the other hand, the execution of a service can generate different outputs and domain effects in different conditions. Hence, the specification in OWL-S allow also the definition of conditional outputs and effects:

```
200 <process:hasResult>
201     <process:Result rdf:ID="startResponsePositive">
202         <process:hasResultVar>
203             <process:ResultVar rdf:ID="requestID">
204                 <rdfs:comment> A unique iID for a Request/Response pair in the given interaction cycle. </rdfs:comment>
205                 <process:parameterType rdf:datatype="&xsd;#anyURI"> &xsd;#string </process:parameterType>
206             </process: ResultVar >
207         </process:hasResultVar>
208         <process:hasResultVar>
209             <process:ResultVar rdf:ID="source">
210                 <rdfs:comment> A URI representing the address of the sender of the event. </rdfs:comment>
211                 <process:parameterType rdf:datatype="&xsd;#anyURI"> &xsd;#string </process:parameterType>
212             </process: ResultVar >
213         </process:hasResultVar>
214         <process:hasResultVar>
215             <process:ResultVar rdf:ID="target">
216                 <rdfs:comment> The URI of the address to which the event will be delivered. </rdfs:comment>
217                 <process:parameterType rdf:datatype="&xsd;#anyURI"> &xsd;#string </process:parameterType>
218             </process: ResultVar >
219         </process:hasResultVar>
```

```
220             <process:hasResultVar>
221                     <process:ResultVar rdf:ID="context">
222                          <rdfs:comment> A unique URI used to identify the current interaction cycle. </rdfs:comment>
223                          <process:parameterType rdf:datatype="&xsd;#anyURI"> &xsd;#string </process:parameterType>
224                     </process: ResultVar >
225             </process:hasResultVar>
226             <process:hasResultVar>
227                     <process:ResultVar rdf:ID="status">
228                          <rdfs:comment> A 'Success' flag </rdfs:comment>
229                          <process:parameterType rdf:datatype="&xsd;#anyURI"> &xsd;#string </process:parameterType>
230                     </process: ResultVar >
231             </process:hasResultVar>
232             <process:hasResultVar>
233                     <process:ResultVar rdf:ID="status">
234                          <rdfs:comment> The current state information. </rdfs:comment>
235                          <process:parameterType rdf:datatype="&xsd;#anyURI"> &xsd;#string </process:parameterType>
236                     </process: ResultVar >
237             </process:hasResultVar>
238             <process:hasResultVar>
239                     <process:ResultVar rdf:ID="data">
240                          <rdfs:comment> The additional arbitrary data. </rdfs:comment>
241                          <process:parameterType rdf:datatype="&xsd;#anyURI"> &xsd;#string </process:parameterType>
242                     </process: ResultVar >
243             </process:hasResultVar>
244             <process:inCondition>
245                 <expr:SWRL-Condition   rdf:ID="hasContent">
246                       <rdfs:label> hasContent( target, Content)  </rdfs:label>
247                       <expr:expressionObject>
248                              <swrl:AtomList><rdf:first>
249                                     <swrl:IndividualPropertyAtom>
250                                           <swrl:propertyPredicate rdf:resource="&this;#hasContent"/>
251                                           <swrl:argument1 rdf:resource="#target"/>
252                                           <swrl:argument2 rdf:resource="#Content"/>
253                                     </swrl:IndividualPropertyAtom> </rdf:first>
254                              <rdf:rest rdf:resource="&rdf;#nil"/>
255                              </swrl:AtomList>
256                       </expr:expressionObject>
257                 </expr:SWRL-Condition>
258         </process:inCondition>
259         <process:hasEffect>
260                 <expr:SWRL-Expression  rdf:ID="updateState">
261                       <expr:expressionObject>
262                              <swrl:AtomList><rdf:first>
263                                     <swrl:ClassAtom>
264                                           <swrl:propertyPredicate rdf:resource="&this;#hasRunningState"/>
265                                           <swrl:argument1 rdf:resource="#MC_state"/>
266                                           <swrl:argument2 rdf:resource="#Running"/>
267                                     </swrl:ClassAtom> </rdf:first>
268                              <rdf:rest rdf:resource="&rdf;#nil"/>
269                              </swrl:AtomList>
270                       </expr:expressionObject>
271                 </expr:SWRL-Expression>
272         </process:hasEffect>
```

This example states that if the Start operation is successful then an acknowledgement StartResponsePositive is given as output and the «world» state is changed. The Modality Component state is changed from Idle to Running (the whole rule reflecting the W3C's MMI protocol specification).

As the example shows, the effect of the service in the «world» only concerns the results in the domain information of the system; in our case, the state of the Modality Component (line 264 to 267).

- DOLCE

DOLCE is a library of upper ontologies [Gangemi et al. 2002] written on OWL. DOLCE has a cognitive orientation: it aims at capturing the ontological categories underlying natural language and human common sense.

According to DOLCE, the most important criteria to conceptualize some reality is the fact that entities can be co-located in the same space-time:

«Commonsense distinguishes between things (spatial objects like houses and computers) and events (temporal objects like bank transfers and computer repairs). In the wake of relativity theory, however, time is viewed as another dimension of objects on a par with the traditional spatial dimensions.

Considering the consequences of this scientific theory (or theories), some philosophers and computer scientists have come to believe that the commonsense distinction between things that are and things that happen should be abandoned in favor of a unified viewpoint.

*According to these **revisionist** researchers, everything extends in space and time, and the distinction between things and events is an (ontologically irrelevant) historical and cognitive accident. (...)*

*A **multiplicative** ontology allows for different entities to be co-localized in the same space-time. These entities are assumed to be different because they have incompatible essential properties.*

This case is often presented through the problem of the vase and the clay it is made of. It seems natural to assume that the vase ceases to exist when a radical change in shape occurs (for instance, when it breaks in peaces). Instead, the amount of clay is not altered by such events.

*According to the **multiplicativist**, these observations show that there must be different (yet related) entities that are co-localized: the vase is constituted by an amount of clay, but it is not an amount of clay. Indeed, when a vase-master shapes a particular amount of clay, new properties are instantiated, and this justifies the emergence of a new entity that we call a vase.*

*This solution is opposed by the **reductionists**, which provide a different answer to this issue. They postulate that each space-time location contains at most one object. Incompatible essential properties (like those that distinguish the vase from the clay) are regarded as byproducts of the different viewpoints one can assume about spatio-temporal entities.*

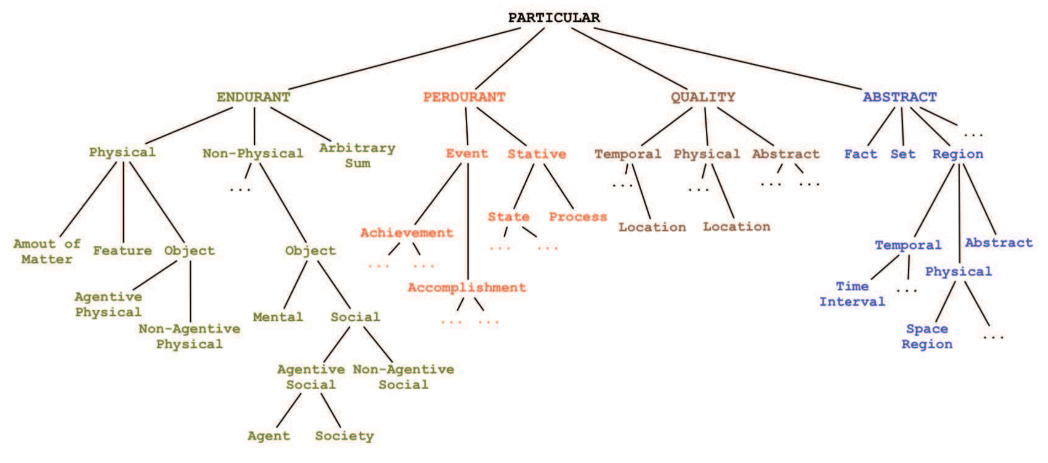
*The vase and the clay are surely different, **reductionists** claim, although not as entities but as views of the same spatio-temporal object. (...)*

*The problem of representing time and modality is an old and ever recurrent quandary in artificial intelligence. Basically, two approaches are possible: either **one includes modal and temporal operators in the formal system from the very beginning**, or reproduces modal reasoning into a first-order language adding time and world (or situation) parameters to the predicates.»* [Gangemi et al. 2002]

DOLCE includes modal and temporal operator as foundations for all knowledge from the very beginning. In this aspect, it is a a «multiplicative» ontology that gives tools to conceptualize spatio-temporal continuous and co-located entities.

DOLCE is presented by its authors as an "ontology of particulars", an ontology of instances, rather than an ontology of universals or properties. The ontology can be described by a taxonomy of categories of particulars with four categories at the higher level explained hereafter:
- Endurant,
- Perdurant,
- Quality
- and Abstract [**Figure 2.2.17**].

An **Endurant**, is an entity that is «**in**» the time. It is «wholly» present (all their proper parts are present) at any time of its existence. For example, a joystick is an endurant. An Endurant (called in Soa2m «discrete entity») exists in a 3D space and is a changing entity in the sense that at different moments Tn it can instantiate different properties. For example, the joystick can pivot in its base changing of orientation in different Tn moments of time.



Basic Taxonomy of DOLCE

In contrast, a **Perdurant**, is an entity that «happens in time»: it is only partially present at each instant. For example, a video.

This is a changing entity in the sense that at different phases it can have totally different properties in its parts (sequences). Perdurants (called in Soa2m «continuous entity») are entities that extend in time by accumulating different 'temporal parts', so that, at any time **Tn** at which they exist, only their temporal parts at **Tn** are present.

For example, at any **Tn** time only the current sequence is present but not the video as a whole. And each sequence can be completely different from the previous and the next sequence being a coherent whole anyway.

For example, a thing that we buy today can be considered as an endurant because at the moment of the purchase (now) it is wholly present, while an «on-demand service» is a perdurant because, the service is never completely present at a single **Tn** moment (now).

So, it is possible to distinguish between «ordinary objects» (like the purchased thing) and «events or process» (like a service).

Endurants can «genuinely» change in time, in the sense that the very same Endurant as a whole can have incompatible properties at different times, while Perdurants cannot change in this sense, since none of their parts keeps the total identity in time.

For example, if «this image» has a property at a time T' «it's white», and a different, incompatible property at time T" «it's yellow»: in both cases we refer to the whole object, without picking up any particular part of it.

On the other hand, when we say that a perdurant «playing a sound» has a property at T' «fast» (during the first five minutes) and an incompatible property at T" «slow» (toward the end of the sound) there are two different parts exhibiting the two different properties.

In DOLCE, the main relation between endurants and perdurants is participation: an endurant «lives» in time by participating in some perdurant(s). For example, a person, which is an Endurant, may participate in a dialog, which is a Perdurant. A person's life is also a Perdurant, in which a person participates throughout its all duration.

A **Quality** can be seen as a basic entity we can perceive or measure: shapes, colors, sizes, sounds, smells, as well as weights, lengths, electrical charges... It is a sensorial notion in DOLCE. No two particulars can have the same quality, and each quality is dependent on the entity it describes. A quality is then an empirical description of things.

The color quality is therefore dependent on the sensorial perception of the form, and two sensorial perceptions are never equal. We express that they have the same color in reference of an interpretation space, in our case, the color space.

Then, each quality type has an associated quality space with a specific structure: lengths are usually associated to a metric linear space, and colors to a topological space.

The structure of these spaces reflects our perceptual and cognitive bias: in DOLCE the notion of ***quale*** designate quality regions, which roughly correspond to the qualitative sensorial perception in humans.

Finally, an **Abstract** do not have spatial nor temporal qualities, and it is not a quality itself.

The taxonomy of the other basic categories of particulars assumed in DOLCE includes, for example, abstract quality, abstract region, agentive physical object, amount of matter, non-agentive physical object, physical quality, physical region, process, temporal quality, temporal region.

The following table shows some examples of occurrences of these DOLCE categories:

Examples of the DOLCE categories

Accomplishment	a conference, a concert, a performance
Achievement	reaching the summit of K2, a departure, a death
Agentive Physical Object	a human person (as opposed to legal person)
Amount of Matter	some air, some gold, some cement
Arbitrary Sum	the association of my left foot and my car
Feature	an opening, a boundary, an edge
Mental Object	a percept, a sense datum

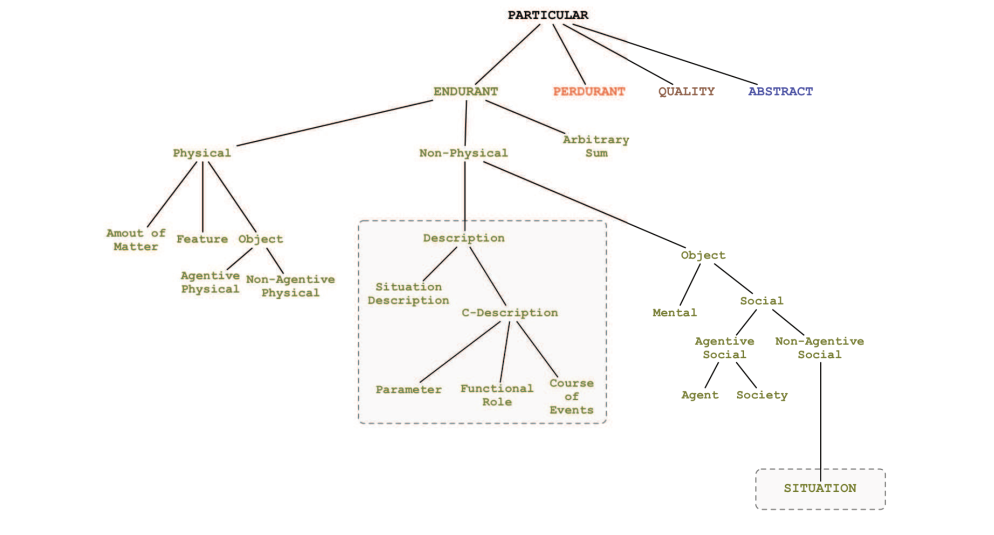
Accomplishment	a conference, a concert, a performance
Achievement	reaching the summit of K2, a departure, a death
Agentive Physical Object	a human person (as opposed to legal person)
Amount of Matter	some air, some gold, some cement
Arbitrary Sum	the association of my left foot and my car
Feature	an opening, a boundary, an edge
Mental Object	a percept, a sense datum
Non-agentive Physical Object	(Thing without will) a computer, a human body, a keyboard
Non-agentive Social Object	(Social Thing without a will) a description, a goal, a task, an activity
Physical Quality	the weight of a pen, the color of a figure, the pitch of a sound
Physical Region	the physical space, an area in the color spectrum, 80Kg
Process	running, writing, selecting, cropping
Social Agent	(Person figure with a will) a (legal) person, a client, a user
Society	(Person group with a common will) Samsung, Apple, the European Bank
State	being running, being open, being idle, being red, being paused
Temporal Quality	(Time interval) the duration of a sequence, the starting time of a presentation
Temporal Region	(Time dimension) the time axis, 22 june 2002, one second, the runtime

On the other hand, the Description & Situations (D&S) ontology is designed as a plug-in to the DOLCE foundational ontology. D&S is intended to provide a framework for representing contexts, methods, norms, theories, situations, and models at first-order, allowing a partial specification of those entities.

D&S axioms try to capture the notion of «situation» as a unifying entity out of a *state of affairs*. The unity is provided by a *description*. A *state of affairs* corresponds to a context, and is a set of assertions coherent with the axioms in a first-order theory, e.g, a trajectory or a multimodal interaction.

A *description* is an entity that partly represents a point of view (or one of its elements) and can be conceived by an agent: either human, collective, social, or artificial, e.g., the visual mode, a media, a recognition, a context of interaction in nomadic uses. It references non-physical objects, as plans, goals, motivations, parameters, mental contents.

In DOLCE D&S a *situation* is constituted by the entities and the relations among them, mentioned in assertions from a *state of affairs*. It is also an entity in the *state of affairs* that partly represents a model for some point of view, according to selected axioms. Then, when a description is applied to a *state of affairs*, the situation structure emerges. Thus, a situation always is in pair with a description, both are non-physical entities, but while a situation is a social object (arising from an agreement) a description is only a non-physical discrete entity (an endurant) which is mostly arbitrary and proposed as an individual point of view (see the figure below).



Taxonomy of D&S Ultralite added to DOLCE

Un modèle Soa multimodal et sémantique et son support pour la découverte et l'enregisrement des services d'assistance

RÉSUMÉ : Les entrées et sorties unimodales dans les systèmes actuels ont atteint une maturité reconnue, avec les applications tactiles ou par les services distribués pour la geo-localisation ou la reconnaissance de la parole, du son ou l’’image. Cependant, l'intégration et l’instanciation de toutes ces modalités, manque d’une gestion intelligente du contexte d’acquisition et de restitution basée sur des notions fortement formalisées mais reflétant le sens commun. Ceci demande un comportement plus dynamique du système avec une approche plus adéquate pour gérer l'environnement de l'utilisateur.

Cependant,la technologie nécessaire pour atteindre un tel objectif n’est pas encore disponible de façon standardisée, tant au niveau des descriptions fonctionnelles des services unimodaux que de leur description sémantique. Ceci est aussi le cas pour les architectures multimodales, où la composante sémantique est produite par chaque projet sans un accord commun pour assurer l’interoperabilité et est souvent limitée au traitement des entrées et sorties ou aux processus de fusion/fission strictement nécessaires au projet.

Pour combler cette lacune, nous proposons une approche sémantique orientée services pour une architecture multimodale générique qui vise à améliorer la description et la découverte des composants de modalité pour les services d'assistance: l'architecture Soa2m. Cette architecture se veut entièrement focalisée sur la multimodalité et enrichie avec des technologies sémantiques car nous croyons que cette orientation permettra d'enrichir le comportement autonome des applications multimodales, avoir une perception robuste des échanges, et contrôler l'intégration sémantique des interactions homme-machine.

Mots clés : Multimodal IHM, Web Sémantique, SOA, Ontologies, Interfaces Utilisateur, Découverte de Services, Adaptation Sémantique, Applications Web Multicanal.

A Soa model, semantic and multimodal, for the discovery and registration of assistance services

ABSTRACT : Unimodal inputs and outputs in current systems have become very mature with touch applications or distributed services for geo-localization or speech, audio and image recognition. However, the integration and instantiation of all these modalities, lack of an intelligent management of the acquisition and restitution context, based on highly formalized notions reflecting common sense. This requires a more dynamic behavior of the system with a more appropriate approach to manage the user environment.

However, the technology required to achieve such a goal is not yet available in a standardized manner, both in terms of the functional description of unimodal services and in terms of their semantic description. To fill this gap, we propose a semantic service-oriented generic architecture for multimodal systems. This proposal aims to improve the description and the discovery of modality components for assistance services: this is the architecture Soa2m. The architecture is fully focused on multimodality and it is enriched with semantic technologies because we believe that this approach will enhance the autonomous behavior of multimodal applications, provide a robust perception of the user-system exchanges, and help in the control of the semantic integration of the human-computer interaction.

Keywords : Multimodal HCI, Semantic Web, SOA, Ontology, User Interfaces, Services Discovery, Semantic Adaptation, Digital Media.