Promoting and characterizing the menu to keyboard shortcuts transition

Emmanouil Giannisakis

▶ **To cite this version:**

Emmanouil Giannisakis. Promoting and characterizing the menu to keyboard shortcuts transition. Human-Computer Interaction [cs.HC]. Université Paris Saclay (COmUE), 2019. English. NNT : 2019SACLT033 . tel-02484851v2

HAL Id: tel-02484851
https://pastel.hal.science/tel-02484851v2

Submitted on 28 Feb 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

université PARIS-SACLAY

TELECOM ParisTech

NNT : 2019SACLT033

Promoting and characterizing the menu to keyboard shortcuts transition

Thèse de doctorat de l'Université Paris-Saclay
préparée à Télécom ParisTech

École doctorale n°580 Sciences et technologies de l'information (STIC)
Spécialité de doctorat : Informatique

Thèse présentée et soutenue à Paris, le 18/07/2019, par

GIANNISAKIS EMMANOUIL

Composition du Jury :

CONVERSY Stéphane Professor, Université de Toulouse	Président
CALVARY Gaëlle Professor, Grenoble Institute of Technology	Rapporteur
CASIEZ Géry Professor, Université de Lille	Rapporteur
TSANDILAS Theophanis Chargé de Recherche - CR1, Inria	Examinateur
BAILLY Gilles CNRS researcher (HDR), Sorbonne Université	Directeur de thèse

Thèse de doctorat

Emmanouil Giannisakis: *Promoting and characterizing the menu to keyboard shortcuts transition,* An Homage to The Elements of Typographic Style, © June 2018

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Favoriser et caractériser la transition des menus vers les raccourcis claviers

Mots clés : IHM, sélection de commandes,transition vers des méthodes expertes

Résumé : Les utilisateurs sélectionnent fréquemment des commandes sur des ordinateurs personnels, des tablettes ou des smartphones. Ils peuvent utiliser des méthodes novices comme les menus linéaires, les bandeaux de commandes ou les barres d'outils. Ils peuvent également utiliser des méthodes expertes telles que les raccourcis clavier ou les raccourcis gestuels qui sont plus efficaces. Cependant, nous observons que de nombreux utilisateurs continuent d'utiliser des méthodes novices, même des utilisateurs expérimentés, parce qu'ils ne parviennent pas à faire la transition de méthodes novices à des méthodes expertes. Cela peut avoir un impact important sur la productivité.

Dans cette thèse, j'étudie la transition des menus vers les raccourcis clavier sur les ordinateurs personnels. Je me concentre sur deux objectifs complémentaires. Le premier objectif consiste à favoriser l'utilisation des raccourcis clavier en concevant de nouvelles techniques d'interaction. Mon approche consiste à (1) identifier les éléments clés d'une commande (nom, icône et raccourci), (2) décrire comment ces éléments sont représentés dans les interfaces traditionnelles et leurs impacts sur la performance et (3) proposer des solutions alternatives. En particulier, je conçois, réalise et évalue deux nouvelles techniques d'interaction qui explorent différents emplacements des raccourcis clavier : la première intègre les raccourcis clavier dans les icônes. La seconde explore la position des raccourcis clavier dans un item de menus. Nous montrons que les deux techniques d'interaction sont prometteuses pour promouvoir l'utilisation des raccourcis clavier. En outre, j'examine comment l'association ¡ nom de la commande - raccourci clavier¿ influence l'utilisation des raccourcis clavier.

Le deuxième objectif consiste à mieux caractériser la transition des menus vers les raccourcis clavier. Je souligne les incohérences entre les caractérisations théoriques et empiriques de cette transition, ce qui rend difficile l'évaluation et la comparaison des techniques d'interaction. Je présente une nouvelle méthodologie pour estimer des marqueurs comportementaux théoriques sur des données empiriques. En particulier, je conçois et évalue un algorithme pour identifier automatiquement le début et la fin de la transition. J'apporte également de nouvelles perspectives sur le comportement des utilisateurs avant, pendant et après la transition.

Cette thèse offre (1) une meilleure compréhension de la transition des menus vers les raccourcis clavier et des facteurs impliqués dans cette transition, (2) de nouvelles méthodes pour caractériser cette transition et (3) deux nouvelles techniques d'interaction pour promouvoir les raccourcis clavier.

Title : Promoting and characterizing the menu to keyboard shortcuts transition

Keywords : HCI, command selection, transition to expert methods

Abstract : Users frequently select commands on personal computers, tablets or smartphones. They can use novice methods such as linear menus, ribbons and toolbars. They can also use expert methods such as keyboard shortcuts and stroke shortcuts that are more efficient. However, we observe that many users continue to use novice methods even experimented users because they fail to make the transition from novice to expert methods. This can have a strong impact on productivity.

In this thesis, I investigate the transition from menus to keyboard shortcuts on personal computers and focus on two complementary objectives. The first objective consists of promoting keyboard shortcut usage by designing novel interaction techniques. My approach consists of (1) identifying the key elements of a command (name, icon and shortcut), (2) describing how these elements are represented in traditional interfaces and their impact on performance and (3) proposing alternative designs. In particular, I design, implement and evaluate two novel interaction techniques exploring different locations of the keyboard shortcut cues: The first one blends the keyboard shortcut cues into command icons. The second one explores the relative position of the keyboard shortcut cues with their corresponding command labels. We show that both interaction techniques are promising for promoting keyboard shortcut usage. In addition, I investigate how the command name - keyboard shortcut mapping impacts keyboard shortcut usage.

The second objective consists of better characterizing the transition from menus to keyboard shortcuts. I highlight inconsistencies between theoretical and empirical characterizations of this transition making difficult to evaluate and compare interaction techniques. I present a novel methodology to estimate theoretical behavioral markers on empirical data. In particular, I design and evaluate an algorithm to automatically identify the beginning and the end of the transition. I also provide new insights regarding users behaviors before, during and after the transition. This thesis offers (1) a better understanding of the transition from menus to keyboard shortcuts and the design factors involved in this transition, (2) novel methods to characterize this transition and (3) two novel interaction techniques to promote keyboard shortcuts.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

PUBLICATIONS

Some ideas and figures have appeared previously in the following publications:

1. Emmanouil Giannisakis, Gilles Bailly, Sylvain Malacria, and Fanny Chevalier. 2017. IconHK: Using Toolbar button Icons to Communicate Keyboard Shortcuts. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI & 17). Association for Computing Machinery, 4715–4726. DOI: https://doi.org/10.1145/3025453.3025595

2. Gilles Bailly, Emmanouil Giannisakis, Marion Morel, Catherine Achard. Caractériser la transition des menus vers les raccourcis claviers. AFIHM. 30eme conférence francophone sur l'interaction homme-machine, Oct 2018, Brest, France. IHM-2018, pp.30-41, 2018, Articles Scientifiques.

ACKNOWLEDGMENTS

It takes a village to complete this thesis, so I would like to dedicate this thesis to all of the people that help to get here.

First, I want to thank Gilles Bailly. I consider myself lucky to have you as a mentor and supervisor. I am particularly grateful for all your patience, your willingness to listen and argue with me, for always being available and for being the biggest supporter of my work.

I would also like to thank all the jury members: Gaëlle Calvary, Géry Casiez, Stéphane Conversy, and Theofanis Tsandilas for agreeing to be part of my thesis. Thank you very much for taking the time out of your busy schedule to read my thesis and provide useful and insightful comments.

Also, I thank Telecom Paris for funding my thesis.

I also want to thank the fantastic people working at INSEAD. Jean-Yves, Huong, Germain, and Sebastien, all of you, were extremely helpful in organizing my studies and to find participants, but also you were a delight to hang out.

A big thanks to my colleagues from Aviz, Ilda, Ex-Situ, and Diva. All of you were great moral support, especially those who participated on Thursday nights at Butte aux Cailles (James, Hugo, Arnaud, etc.).

Coming to a new country to start a Ph.D. thesis is not easy, but some special people made this whole experience beautiful. Bruno, Abby, Marc and Manos, I couldn't wish for better friends. I will always remember and cherish all the things we did together: our trips (US, Helsinki, Mulhouse, and the great city of Nantes), our drinking nights and our walks around Paris. Thank you for all of these great memories.

I spent three wonderful years living with two amazing people. Achille, you are the best roommate that I ever had, you made our every day lives more enjoyable. Anthi, thanks for being here for me for all these years. I could always count on you. You were still there to support me and make me feel better when I was feeling down, and you were more happy for my successes than I was. I am so fortunate to have a friend like you.

Finally, I want to thank my family and especially my parents. I wouldn't be the person I am without your constant support, encouragement, and unconditional love. But most importantly, thank you for providing me all the necessary tools to be able to be part of this world.

CONTENTS

1 INTRODUCTION 1
1.1 Research Goals . . . 2
1.2 Approach . . . 2
1.3 Outline . . . 3
1.4 Contributions . . . 4

I RELATED WORK

2 HUMAN FACTORS FOR TRANSITION TO MORE EFFICIENT METHODS 7
2.1 Performance improvement while using one method . . . 7
2.1.0.1 Theories . . . 8
2.1.0.2 Mathematical models . . . 8
2.2 Transitioning to a new method . . . 10
2.2.1 Characterizing transition by "State" . . . 10
2.2.2 Characterizing transition by adoption of the new method 11
2.2.2.1 Theories . . . 11
2.2.2.2 Mathematical models . . . 11
2.2.3 Characterizing transition by performance improvement . 14
2.2.3.1 Theories . . . 15
2.2.3.2 Mathematical models . . . 15
2.2.4 Discussion . . . 16
2.3 Favoring suboptimal methods . . . 17
2.4 Factors for promoting efficient methods . . . 18
2.4.1 Initial Switch to a new method . . . 18
2.4.2 Performance Dip . . . 19
2.5 The Role of Learning in transition . . . 19
2.5.1 Explicit learning . . . 19
2.5.2 Implicit Statistical Learning . . . 20
2.6 Implications for system design . . . 21
2.6.1 Feedback and Feedforward . . . 21
2.6.2 Contextual cueing . . . 21
2.7 Conclusion . . . 23
3 COMMAND SELECTION METHODS : DESIGN AND INTERACTION TECHNIQUES 25
3.1 Current command selection methods . . . 25
3.1.1 Recognition-Based methods . . . 25
3.1.1.1 Method Characteristics . . . 26
3.1.1.2 Design Guidelines . . . 27
3.1.2 Recall-Based methods . . . 27
3.1.2.1 Method Characteristics . . . 28
3.1.2.2 Design Guidlines . . . 29
3.1.3 Command selection methods and user behavior . . . 29
3.1.3.1 Intramodal performance improvement . . . 30

3.1.3.2 Forgetting and Relearning . . . 30
3.1.3.3 Method adoption . . . 31
3.1.3.4 Intermodal performance improvement . . . 31
3.1.3.5 Discussion . . . 31
3.2 Interface design practices for command selection methods 32
3.2.1 Design practices . . . 32
3.2.1.1 Practices for designing each element . . . 32
3.2.1.2 Practices for layout design (interface design) . . . 33
3.2.2 Relationship between the elements . . . 36
3.2.2.1 Command name - Command icon . . . 36
3.2.2.2 Command name - Keyboard shortcut . . . 37
3.2.2.3 Command icon- Keyboard shortcut . . . 38
3.2.3 Discussion . . . 38
3.3 Interaction Techniques to promote shortcut usage . . . 39
3.3.1 Learning the Shortcut . . . 39
3.3.2 Raising Awareness . . . 43
3.3.3 Perception of Future Performance . . . 43
3.3.4 Discussion . . . 44
3.4 Conclusion . . . 44

II REVISITING CURRENT LAYOUT DESIGN PRACTICES

4 EXPLORING THE COMMAND ICON – KEYBOARD SHORTCUT RELATIONSHIP 47
4.1 The IconHK approach . . . 48
4.1.1 Embedding keyboard shortcuts into toolbar buttons . . . 48
4.1.1.1 Conveying the hotkey . . . 48
4.1.1.2 Embedding Modifiers . . . 50
4.1.2 The magnification continuum of IconHK . . . 51
4.1.2.1 Formal definition . . . 52
4.1.2.2 Magnifying along the continuum: symbol saliency . . . 53
4.1.3 IconHK dynamic behavior . . . 54
4.1.4 Mnemonic mechanisms . . . 56
4.2 Redesign of the Photoshop palette . . . 56
4.3 Study 1: IconHK as a mnemonic aid . . . 57
4.3.1 Participants . . . 58
4.3.2 Design . . . 58
4.3.3 Procedure . . . 58
4.3.4 Results . . . 59
4.4 Study 2: Insights from designers . . . 60
4.4.1 Participants . . . 60
4.4.2 Procedure . . . 60
4.4.3 Results . . . 61
4.4.4 Follow up: Focus on Positive and Negative space . . . 63
4.4.5 Discussion . . . 63
4.5 IconHK Maker . . . 64
4.6 Conclusion . . . 65
5 EXPLORING THE COMMAND NAME – KEYBOARD SHORTCUT RELATIONSHIP 67

5.1 Revisiting the command name - keyboard shortcut mapping 68
5.2 Revisiting the linear menu layout 69
5.2.1 Current approaches to decrease the distance between the keyboard shortcut and command name 69
5.2.2 Design space dimensions: position & alignment 70
5.2.3 Exploration of the design space 71
5.3 Experiment 1 71
5.3.1 Participants 71
5.3.2 Apparatus 72
5.3.3 Experimental design 72
5.3.4 Procedure 74
5.3.5 Design 74
5.3.6 Results 75
5.3.6.1 Keyboard shortcuts adoption 75
5.3.6.2 Errors and time 76
5.3.6.3 Perception of past performance 77
5.3.7 Discussion 77
5.4 Experiment 2 78
5.4.1 Participants 78
5.4.2 Apparatus 78
5.4.3 Experimental design 79
5.4.4 Procedure 80
5.4.4.1 Design. 80
5.4.5 Results 81
5.4.5.1 Keyboard shortcut adoption per layout 81
5.4.5.2 Keyboard shortcut adoption per command length . . . 81
5.4.5.3 Errors and Time 82
5.4.5.4 Participant inclusion criteria for data-analysis 82
5.4.6 Discussion 84
5.5 Experiment 3 85
5.5.1 Participants 86
5.5.2 Apparatus 86
5.5.3 Experimental design 87
5.5.3.1 Frequency 87
5.5.3.2 Stimulus & Task 87
5.5.4 Procedure 88
5.5.5 Results 90
5.5.5.1 Keyboard shortcut adoption 90
5.5.5.2 Keyboard shortcut adoption per user profile 91
5.5.6 Discussion 92
5.6 Conclusion 93

III CHARACTERIZING THE MENU TO KEYBOARD SHORTCUT TRANSITION

6 CHARACTERIZING MENU TO KEYBOARD SHORTCUT: TRANSITION START, TRANSITION DURATION AND PERFORMANCE 97
6.1 Current characterization of transition 97

6.1.1 Theoretical Characterization . . . 97
6.1.1.1 Characterization from the definition . . . 97
6.1.1.2 Characterization from the theoretical frameworks . . . 98
6.1.1.3 Characterization from the mathematical models . . . 99
6.1.1.4 Discussion . . . 100
6.1.2 Empirical characterization . . . 100
6.1.2.1 Data Aggregation . . . 101
6.1.2.2 Metrics . . . 101
6.1.2.3 Data cleaning . . . 102
6.1.2.4 Discussion . . . 104
6.1.3 Discussion . . . 104
6.2 Expanding current theoretical markers . . . 105
6.3 Case study: Grossman et al. user study . . . 106
6.3.1 Study overview . . . 106
6.3.1.1 Experimental plan . . . 106
6.3.1.2 Summary of results . . . 107
6.4 Data annotation . . . 108
6.4.1 Problem formulation . . . 108
6.4.2 Manual annotation . . . 109
6.4.2.1 Experimental plan . . . 109
6.4.2.2 Annotators Agreement . . . 110
6.4.2.3 Resolution of disagreements . . . 111
6.4.2.4 Validity of annotations . . . 111
6.4.2.5 Discussion . . . 112
6.4.3 Automated annotation . . . 112
6.4.3.1 Sliding window . . . 112
6.4.3.2 Machine learning algorithms . . . 117
6.4.3.3 Evaluation of the algorithms . . . 117
6.4.3.4 Discussion . . . 119
6.5 Detailed analysis of the transition . . . 120
6.5.1 Data and analysis technique . . . 120
6.5.2 Results . . . 121
6.5.2.1 Behavioral markers . . . 121
6.5.2.2 Additional Analysis . . . 125
6.5.3 Discussion . . . 126
6.6 Conclusion . . . 128

IV GENERAL DISCUSSION AND CONCLUSIONS

7 CONCLUSION 131
7.1 Summary . . . 131
7.2 Future work and perspectives . . . 133
7.2.1 Designing interfaces that support transition . . . 133
7.2.2 Understanding and characterizing the transition . . . 136

V APPENDIX

H APPENDIX ICONHK 141

I APPENDIX: KEYBOARD SHORTCUTS IN MENU 143
1.1 Designer Responses 143
1.2 Experiment 1: further analysis 144
1.3 Experiment 2: further analysis 146
1.4 Command - Keyboard shortcut mapping material 149
J APPENDIX TRANSITION 153
J.1 Additional analysis of menu item length - Experiment 2 . . 153
J.1.1 BASELINE 153
J.1.2 RIGHT 154
J.2 Additional analysis of menu layout - Experiment 3 155
J.2.1 FAMILIAR 155
J.2.2 NOT-FAMILIAR 156
J.3 Additional analysis of menu item length - Experiment 3 . . 157
J.3.1 BASELINE 157
J.3.2 LEFT 158
J.3.3 Time signal encoder-decoder with a 1D convolutional layer 158
J.3.3.1 Problem formulation and related work 158
J.3.3.2 Phase extraction: the encoder-decoder 159
J.3.3.3 Estimation of the beginning and end of the transition . 159

BIBLIOGRAPHY 161

LIST OF FIGURES

Figure 2.1 Both models can capture the performance improvment eq. (2.1) 9
Figure 2.2 The constant model takes no parameters and it is suitable for the cases where users do not transition 12
Figure 2.3 The constant-constant model takes one parameter θ which indicates the transition point from one method to the new one. 12
Figure 2.4 Sigmoid functions can model the gradual transition. They usually have three parameters that can provide useful insights: α which is the upper asymptote , μ which is the inflection point i.e. the 50% of *alpha* and σ the speed of the raise to α 13
Figure 2.5 The saturating exponential function models also the gradual transition 14
Figure 2.6 Intermodal performance improvement [39] . . 16
Figure 3.1 Recognition-Based methods: (A) Linear menu, (B) Toolbar, (C) Ribbons, (D) Keyboard shortcuts, (E) Command line interface and (F) Stroke shortcuts. 26
Figure 3.2 Recall-Based methods methods: (A) Keyboard shortcuts, (C) Command line interface and (B) Stroke shortcuts. 27
Figure 3.3 An example of access keys [178]. 28
Figure 3.4 An example of a convectional menu layout of the inkscape application. 34
Figure 3.5 Two examples of toolbar layout. The top layout shows the toolbar from the Pycharm application. This toolbar shows only items and when the users hover over a button the name of the command along with the keyboard shortcut appears. The bottom layout shows the toolbar from Keynote application. This toolbar displays icons along with text for each command. When the users hover over an item a tooltip appears informing them about the functionality of the command 36

Figure 3.6 The analysis of the relations between the three elements highlighted that command name and command icon have a solid relationship (annotated with black solid line) while there are still opportunities to improve the relationship between the Keyboard shortcut (KS) and the other two elements (annotated with the red dashed line). . 37

Figure 3.7 (A) ExposeHK [107] overlays the KS on top of the RECOGNITION-BASED methods when a modifier is pressed, (B) Keycue [157] displays a pop-up window with the relevant KS when a modifier is pressed, (C) Marking menus [95] display a pie menu in novice mode to help users learn the gesture. 40

Figure 3.8 (A) Xerox Star keyboard [24] adds extra buttons for common commands like Copy, (B) GestAKey [146] keyboard can recognize gestures , (C) Ctrl-Mouse [127] adds two extra buttons in the mouse while ToucCtrl [127] recognizes the Ctrl modifier when the users place their hand on the mouse, (D) Métamorphe [17] augments the key functionality so it can support more ways to input a command, (E) Optimus keyboard [11] and (F) Touch display keyboards [27] have keys that contain small screens which can be used to promote keyboard shortcuts 41

Figure 3.9 (A) FingerSense [175] recognizes the finger that press the button and executes the appropriate command, (B) Finger aware shortcuts [188] detect the hand , the finger the posture , (C) [30] can recognize wrist rotation 41

Figure 3.10 (A) Flower Menus [12] can support up to 56 commands, (B) Wavelet menus [55] create a visual perception of menu hierarchy , (C) Polygon [186] and (D)Zone menus [186] increase the breadth of the menu by considering the orientation and the starting point of the stroke, (E) GestKeyboard [185] supports strokes on the keyboard, (F) Zhao et al. [187] proposed a variation of marking menus to support hierarchical lists 42

Figure 3.11 (A) Grossman et al. [77] proposed several menu designs to promote keyboard shortcuts, (B) Hotkey Coach [94] shows a transparent window that contains the KS when the users select a command using the menu or the toolbar 43

Figure 3.12 Skillometers [108] informs the users about their performance with the current method they use and the potential performance with other methods 43
Figure 4.1 This chapter proposed a novel design perspective to design command icons that visual blend the KS cue . The result icons can then communicate the command name along with the KS (*semantic relationship improvement*. Furthermore, the resulted icon brings together the KS cue with the command icon (*spatial relationship improvment*) and maximizes it's exposure. 47
Figure 4.2 Empty space leverages the blank space in or around the pictograph to display the letter 49
Figure 4.3 Positive space leverages the silhouette or the most salient features (i.e. edges) of the pictograph to display the letter 49
Figure 4.5 49
Figure 4.6 Negative space the open space around an object to disguise a letter 50
Figure 4.7 Example of icons augmented with iconHK that embed modifiers using the keyboard-location inspired mapping: alt at the bottom-right, Ctrl at the bottom-left, shift at the top-left. The commands *equal line space* (left) and *RGB colors* (right) respectively have Ctrl + E and Ctrl + alt + E as keyboard shortcuts. Corners are filled in dark to convey modifiers involved in the shortcuts (bottom-left corner only for *equal line space*, two bottom corners for *RGB colors*). The bottom-left corner is highlighted in blue when Ctrl is pressed to provide feedback. 51
Figure 4.8 Variations of toolbar buttons on the IconHK continuum for the *Pencil*, *Move* and *Expand vertically* commands. From 0 to 1 on the continuum: *Pencil* scales down while a W scales in, exploiting the empty space; *Move* rotates and fades out the pointer while revealing a A overlaying the edges; *Expand vertically* changes the background color of the button to emphasize the H symbol in the negative space. 52

Figure 4.9 Example of a toolbar using IconHK. Top: by default ($M_{\lfloor}$), the magnification of all buttons is equal to $M_{\sqsubset}$: the hotkey symbol is not emphasized. Bottom: the individual $M_{\lfloor}$ of each button reflects user's command selection habits: the more frequently a command is selected by clicking, the higher the value of $M_{\lfloor}$ is. 54
Figure 4.10 When the user presses a modifier key, the magnification factor of all buttons increases to $M_{\sqsupset}$ to maximize the saliency of the hotkey when the user needs them the most. 55
Figure 4.11 When the user presses on a toolbar button, its saliency instantly increases to $M_{\sqsupset}$. 55
Figure 4.12 Possible adaption of the Adobe Photoshop CC 2015 interface using IconHK. Top: default toolbar. Middle & Bottom: the adapted toolbar integrating IconHK principles at two different levels of magnification. All buttons embed their associated hotkey, except the Blur button (water drop) which lacks a hotkey. 57
Figure 4.13 Icons used study 1. Top: icons used for the trainings (magnified for IconHK); Bottom: icons used for the tests (not magnified). 58
Figure 4.14 Percentage of recognition rates for each Icon family, for every block and retention tests (dashed) 59
Figure 4.15 Left: Icons implementing the Empty space strategy from the first part of the study. Right: Positive and Negative sketches with different magnification from the follow-up study: *Plot*: A pie chart pictograph embedding either a 'Y' or a 'K'. *Edit Path*: A connected line embedding either a 'V'or a 'W'. 61
Figure 4.16 The algorithm to identify an hotkey symbol in the positive and negative space. 65
Figure 5.1 In this chapter I first explore how the command name can better communicate the KS (*semantic relationship improvement* by considering the letter of the command name and the KS hotkey. Afterwards I explore different placements of the KS cue in the linear menu that aims to bring closer the command label and the KS cue *spatial relationship improvement*. 67

Figure 5.2 Design space for placing the keyboard shortcut cue (ks) in relation to its command name (cmd) regarding two dimensions: *position* and *alignment*. Shortcut positions: left (c,f), right (a,b,d,e) bottom (h), up (g) or diagonal (not shown in the figure). Shortcut alignments: left (b,h), right (a,c), and centered (f,g). The circled letters highlight the designs that we finally compared in Experiment 1. 70

Figure 5.3 *(A)*: Mean rate of *correct keyboard shortcuts* per mapping quality. *(B)*: Mean rate difference of *correct keyboard shortcuts* between command-keyboard shortcut mapping. All error bars are 95% CIs. . 75

Figure 5.4 *(A)*: Mean rate of *correct keyboard shortcuts* per mapping quality. *(B)*: Mean rate difference of *correct keyboard shortcuts* between mapping qualities. All error bars are 95% CIs. 75

Figure 5.5 Mean rate difference of *correct keyboard shortcuts* between mapping qualities for each command-keyboard shortcut mapping. All error bars are 95% CIs. 76

Figure 5.6 Mean number of errors per layout for each error type. All error bars are 95% CIs. 77

Figure 5.7 Mean number of errors difference per layout for each error type. All error bars are 95% CIs. . . . 77

Figure 5.8 Mean perception of performance (in seconds) for menu and keyboard shortcut for each block. In the background we also show the % mean rate of *correct keyboard shortcuts* per block 77

Figure 5.9 *(A)*: Mean rate of *correct keyboard shortcuts* per layout. *(B)*: Mean rate difference of *correct keyboard shortcuts* between layouts. All error bars are 95% CIs. 81

Figure 5.10 *(A)*: Mean rate of *correct keyboard shortcuts* per command length. *(B)*: Mean rate difference of *correct keyboard shortcuts* between command lengths. All error bars are 95% CIs. 81

Figure 5.11 Mean rate of *correct keyboard shortcuts* per command item length for the BASELINE menu layout. All error bars are 95% CIs. 82

Figure 5.12 Mean rate of *correct keyboard shortcuts* per command item length for the RIGHT menu layout. All error bars are 95% CIs. 82

Figure 5.13 Mean rate of *correct keyboard shortcuts* per layout for each threshold. All error bars are 95% CIs. . 83

Figure 5.14 Mean rate difference of *correct keyboard shortcuts* between layouts for each threshold. All error bars are 95% CIs. 83
Figure 5.15 Mean rate of *correct keyboard shortcuts* per command name lengths for each threshold. All error bars are 95% CIs. 84
Figure 5.16 Mean rate difference of *correct keyboard shortcuts* between command name lengths for each threshold. All error bars are 95% CIs. 84
Figure 5.17 86
Figure 5.19 (A) Overview of the new interface and (B) the 4 stages for each target item. 87
Figure 5.20 % Participants who started the transition before the first tip and after the first, second and third tip respectively. 90
Figure 5.21 *(A)*: Mean rate of *correct keyboard shortcuts* per layout. *(B)*: Mean rate difference of *correct keyboard shortcuts* between layouts. All error bars are 95% CIs. 90
Figure 5.22 Mean rate of *correct keyboard shortcuts* per command item length for the BASELINE menu layout. All error bars are 95% CIs. 91
Figure 5.23 Mean rate of *correct keyboard shortcuts* per command item length for the RIGHT menu layout. All error bars are 95% CIs. 91
Figure 5.24 # of participants who were FAMILIAR and NOT-FAMILIAR with keyboard shortcuts for both experiments. 92
Figure 5.25 *(A)*: Mean rate of *correct keyboard shortcuts*. *(B)*: Mean rate difference of *correct keyboard shortcuts*. Black CIs indicate differences between the two layouts for each user profile (FAMILIAR and NOT-FAMILIAR). 92
Figure 6.1 The three behavioral markers derived from Cockburn et al. [39] and Gray et al. [72] 98
Figure 6.2 An example of the three behavioral markers that are derived from the transition models 99
Figure 6.3 103
Figure 6.4 An updated version of fig. 6.1. We expanded the current markers by changing the *initial switch* marker to beginning and duration of the transition and by adding the rebound duration . . 105
Figure 6.5 Results of the study presented in [77] . Use of keyboard shortcuts by technique and frequency with 95% confidence intervals as well as block and technique selection time. 107

Figure 6.6 Interface for annotating a user's order. A selection is represented by point. The color indicates strategy and the symbol indicates success. Vertical lines indicate the beginning and end of the transition 110
Figure 6.7 Example of how the sliding window works. In this example we used a window size of 4 sequence points and an acceptance threshold of 50% 113
Figure 6.8 (A) Beginning of the transition measured in number of encounter with the command per technique before the beginning of the transition. (B) Differences of the beginning of the transition between the technique. 122
Figure 6.9 (A) Beginning of the transition measured in number of encounter with the command per command appearance frequency before the beginning of the transition. (B) Differences of the beginning of the transition between the command appearance frequencies 122
Figure 6.10 (A) Duration of the transition measured in number of encounter with the command per command appearance frequency. (B) Differences of the duration of the transition between the command appearance frequencies. 123
Figure 6.11 A) Duration of the transition measured in number of encounter with the command per command appearance frequency. (B) Differences of the duration of the transition between the command appearance frequencies. 123
Figure 6.12 (A) Performance gain for each technique measured in seconds. (B) Difference of performance gain for each technique. 123
Figure 6.13 (A) Performance drop for each technique measured in seconds. (B) Difference of performance drop for each technique. 124
Figure 6.14 **(A)**: Number of command encounters between the best time before the beginning of the transition and the performance when the transition started. (B) Mean differences. 124
Figure 6.16 Representation of all command sequences where the user has started a transition. The sequences are aligned with the transition date. Color encodes strategy 125

Figure 6.15 (A) Rebound duration per technique. (B) Mean difference of rebound duration between the techniques. 125
Figure 6.17 Time distribution before the beginning and after the end of the transition for menus and KS (A) and during the transition(B). 126
Figure 7.1 In this thesis we focused on improving the semantic and graphic relationship between the KS and the other two elements (red dotted line). . 131
Figure 7.2 **Left**: IconHK (chapter 4 aims at improving the relationship between the command icon and the KS. **Middle**: The exploration of the position of the KS cue (chapter 5) aimed to improve the graphic relationship between the command name and the KS. **Right**: The exploration of the command name-KS mapping aimed at improving the semantic-syntactic relationship between the command name and the KS. 132
Figure 7.3 To characterize the transition (chapter 6 we used the following markers: 1. *Beginning of the transition*, 2. *Duration of the transition*, 3. *Performance dip*, 4. *Performance gain* and 5. *Rebound duration* . 133
Figure 7.4 (A) Reimagining the photoshop palette using IconHK icons (for more details see chapter 4). (B) Reimagining the Pycharm linear menu by positioning the KS cue on the left of the command 136
Figure h.1 Icons that D1 produced 141
Figure h.2 Icons that D2 produced 141
Figure h.3 Icons that D3 produced 142
Figure h.4 Icons that D2 produced 142
Figure i.1 The menu layouts that the designers had to evaluate. 143
Figure i.2 Designer responses for all menu layouts 144
Figure i.3 Designer responses for BASELINE, LEFT, RIGHT and BELOW layouts 144
Figure i.4 Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs. 145
Figure i.5 Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs. 145
Figure i.6 Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs. 145

Figure i.7 Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs. 145
Figure i.8 Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs. 146
Figure i.9 Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs. 146
Figure i.10 Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs. 147
Figure i.11 Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs. 147
Figure i.12 Mean rate of *correct keyboard shortcuts* per layout for each threshold for BELOW. All error bars are 95% CIs. 147
Figure i.13 Mean rate difference of *correct keyboard shortcuts* between between menu item lengths for each threshold for BELOW. All error bars are 95% CIs. 147
Figure i.14 Mean rate of *correct keyboard shortcuts* per layout for each threshold for LEFT. All error bars are 95% CIs. 148
Figure i.15 Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for LEFT. All error bars are 95% CIs. 148
Figure i.16 Mean rate of *correct keyboard shortcuts* per layout for each threshold for RIGHT. All error bars are 95% CIs. 148
Figure i.17 Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for RIGHT. All error bars are 95% CIs. 148
Figure j.1 Akaike Information Criterion (AIC) analysis indicates that the most suitable model is the one where the α value is different between the models 153
Figure j.2 (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot. . 153
Figure j.3 AIC analysis indicates that the most suitable model is the one where the α value is different between the models 154
Figure j.4 (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot. . 154

Figure j.5 AIC analysis indicates that the most suitable model is the one where the α value is different between the models 155
Figure j.6 (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot. . 155
Figure j.7 AIC analysis indicates that the most suitable model is the one where the α value is different between the models 156
Figure j.8 (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot. . 156
Figure j.9 AIC analysis indicates that the most suitable model is the one where the α value is different between the models 157
Figure j.10 (A) Regression results for each menu item length. (B) since the model. (C) Residual plot. 157
Figure j.11 AIC analysis indicates that the most suitable model is the one where the α value is different between the models 158
Figure j.12 Encoder-decoder of time-series from convolutional neural networks. 158

LIST OF TABLES

Table 6.1 Transition detection rate (in %) in cross-validation Leave one out (LOO) and Leave one subject out (LOSO) with sliding window algorithm, encoder/decoder modeling or HMM. The results are given in the form of the beginning of transition / end of transition. 119
Table 6.2 Number of commands where the user started and completed the transition by technique and frequency. In our analysis we considered the commands with frequencies that are annotated with black color. 121
Table i.1 Set 1 149
Table i.2 Set 2 150
Table i.3 Set 3 151
Table j.1 154
Table j.2 155
Table j.3 156

Table j.4 156
Table j.5 157

LISTINGS

ACRONYMS

GUI	Graphical User Interface
KS	Keyboard shortcut
SS	Stroke Shortcut
CLI	Command Line Interface
ISL	Implicit Statistical Learning
LOO	Leave one out
LOSO	Leave one subject out
AIC	Akaike Information Criterion
HCI	Human computer interaction

1 INTRODUCTION

"Your life does not get better by chance, it gets better by change." — Jim Rohn

We often face situations where we should change our behavior for performance [39, 72, 143], health [129, 134] or environmental [44, 181] reasons. Several of these situations require individuals to be aware of the different alternatives and to have the skills to change their behavior. For instance, high jump athletes had to stop practicing the Scissor method that leads to suboptimal performance and practicing the Fosbury Flop technique [43]. An individual can also learn and practice everyday exercises (e.g. walking) to promote health. How behavior design can shape individual habits and favors behavioral changes?

Different research fields (decision making, cognitive psychology, behavioral economics, etc.) investigate this broad question. In this thesis, I address the problem of performance in Graphical User Interface (GUI) within the context of Human computer interaction (HCI). More precisely, I investigate how interface design can influence the methods the users will choose for selecting commands in GUIs.

HCI is the science that studies the action-perception loop between one or several users and one or several interactive systems. HCI is also the research field focusing on the design, implementation, and evaluation of interactive systems based on human factors.

Selecting commands is one of the most common tasks in HCI. It concerns a variety of users using systems varying from the smartwatch to a wall-size display through desktop workstations and smartphones. Currently, millions of users spend several hours per day to execute commands or applications. For example, a graphic designer select commands to design a logo or a developer to write a chunk of code to complete a new feature for a system. Consequently, saving even a few seconds off of this operation can result in significant cumulative savings when considering a complex task (e.g. photo edition) with strong implications on individual and societal productivity and entertainment [69, 70].

Current GUIs offer multiple methods to select commands. For instance, novice methods such as linear menus, toolbars, ribbons are useful as users with no prior experience can quickly start using them because these methods rely on recognition [39, 119]. Expert methods such as keyboard shortcuts and stroke shortcuts [7, 133] are generally more efficient and target more experienced users as they often rely on recall.

While expert methods are more efficient (they can be two times faster than novice methods [7, 133]), many users fail to make the transition

from novice to expert methods. This behavior has been observed even with users that have many years of experience [25, 26]. Indeed, one limitation of the novice methods is to "trap" users in a sub-optimal plateau of performance [13, 32, 39, 71], which can have a strong impact on productivity. One main challenge in command selection is thus to change users' behavior so that they use more efficient expert methods.

This thesis addresses this challenge and investigates the transition from menus to keyboard shortcuts on personal computers.

1.1 RESEARCH GOALS

The goal of this thesis is to study how interface design changes users' behavior regarding the adoption of more efficient (but effortful) methods in the context of command selection. I refine this general goal into three more concrete subgoals:

RG 1: *Explore the design space of keyboard shortcuts to identify potential for designs.* The goal is to identify the key elements of a command (name, icon and shortcut) and the design factors that can influence keyboard shortcut usage.

RG 2: *Favoring keyboard shortcuts usage.*This goal consists of designing novel interaction techniques that changes users' behavior to promote keyboard shortcuts usage.

RG 3: , This goal will result in new approaches on how to analyze empirical data to better characterize the transition in order to better evaluate and compare interaction techniques.

1.2 APPROACH

I adopt several approaches in this thesis:

THEORETICAL APPROACH. I build on concepts, models and frameworks in cognitive psychology to (1) better understand user expertise and decision making in GUIs and (2) define the key behavioral markers involved in the transition from menus to keyboard shortcuts.

DESIGN APPROACH. This approach consists of exploring the design space [150] of possible interaction techniques. In my context, it consists of identifying interactive mechanisms involved in command selection. It also consists of proposing novel designs that favor the transition from menus to keyboard shortcuts.

EMPIRICAL APPROACH. I conduct controlled laboratory experiments to collect empirical data. Then I analyze these data to understand the impact of interface designs on users' behavior and validate the performance of the proposed interaction techniques.

Methodological approach. I propose novel methods and tools to (automatically) characterize the transition from menus to keyboard shortcuts. This is useful to evaluate and compare the different interaction techniques more precisely.

1.3 OUTLINE

The rest of this thesis is organized as follows:

Chapter 2 **Background on transition to more optimal methods.** In this chapter, I review related work about the transition from one method to a more efficient one. I start by reviewing theories and models in cognitive psychology and HCI characterizing this transition. I then review factors that may inhibit or promote users to transition to more efficient methods. Finally, I focus on the role of learning in the transition by distinguishing conscious and unconscious learning. The main outcome of this chapter is the discussion of theories and models in cognitive psychology that inspire me to design novel interaction techniques in chapters 4 and 5 and to characterize the transition from menus to keyboard shortcuts in chapter chapter 6.

Chapter 3 **Background on command selection methods.** In this chapter, I review related work on the command selection methods that appear in current GUI. First I define, describe and discuss the different characteristics of methods for selecting commands. I then highlight three key elements of command selection (command name , command icon, shortcut) as well as the relations between these elements at the semantic [136, 147] and at graphical level. I demonstrate that current design practices generally do not communicate these different relations to the users offering potential for designs that I exploit in chapters 4 and 5. Finally, I discuss recent interaction techniques promoting keyboard shortcut usage.

Chapter 4 **Exploring the relationship between the command icon and the shortcut (RS 1).** This chapter investigates one relation identified in the previous chapter: the one between the command icon and the keyboard shortcut. I study how the graphic and the semantic relationship between the command icon and the KS can promote KS usage. To achieve this, I propose IconHK a novel icon design approach that visually blends the KS with the command icon. I present different strategies, interaction techniques and examples. I also report on two studies focusing on how users can benefit from the IconHK approach and on designers practices. Finally, I present a tool that aims to accelerate the design of IconHK icons by proposing possible placements of the KS inside the icon.

Chapter 5 **Exploring the relationship between the command name and the KS (RS 1).** This chapter investigates a second relation identified in chapter chapter 3: the relation between the command name and the KS. I first explore the quality of a keyboard shortcut for a given command (semantic level). I then explore the different positions of the keyboard shortcut cue in linear menus (graphical level). We report on three user studies investigating the impact of this relation on keyboard shortcut adoption. Results indicate that this relation can promote keyboard shortcut usage, but the magnitude of the effect remains unclear.

Chapter 6 **Analyzing data to capture the transition to more efficient methods (RS 2).** In this chapter, I highlight inconsistencies between theoretical and empirical characterizations of the transition from menus to keyboard shortcuts. I also highlight the limitations of existing methods to evaluate and compare interaction techniques promoting keyboard shortcuts. I present a novel methodology to estimate theoretical behavioral markers on empirical data. In particular, I design and evaluate an algorithm to automatically identify the beginning and the end of the transition. I also provide new insights regarding users behaviors before, during and after the transition.

Chapter 7 **Conclusion.** In this chapter I provide a summary of this thesis' contribution as well as its limitations and future work.

1.4 CONTRIBUTIONS

To summarize this thesis makes the following contributions:

- *IconHK*: a novel design approach to design command icons that visual blends the KS with the command icon. This work has been accepted in CHI 2017.
- *Revising the linear menu layout*: an exploration of how the position of the semantic and the spatial relationship between the command name and the KS can affect the KS adoption. This work is going to be submitted in CHI2020.
- *Characterizing the transition to KS*: a novel methodology to analyze empirical data that better captures the characteristics of the transition. Part of this work has been submitted in the IHM2018 and will be submitted in the TOCHI journal

Part I

RELATED WORK

2 HUMAN FACTORS FOR TRANSITION TO MORE EFFICIENT METHODS

This chapter offers an overview on the related work regarding transition. I want to start by defining what is transition. According to oxford dictionary transition is "the process or a period of changing from one state or condition to another"[1]. This definition is high level so to make it more specific for my research questions I want to discuss what is *state*.

In this thesis I focus on two different types of *state*. The first type refers to the psychomotor states that users go through while learning how to use a method which I am going to discuss more on this in section 2.1. The second type refers to the method that users use to perform a task and the transition to a more efficient one. I am going to discuss this type in section 2.2.

Another thing I want to mention is that transition is a complex topic of discussion and there are different theories that are related to this topic which can also overlap. In this chapter and in this thesis in general I approach this problem primarily from the users' performance perspective and although I am going to discuss factors that affect users decision to choose a method and to switch to a more efficient one I do not focus on decision making or behavioral economic theories.

Finally another topic I am going to discuss in this chapter is the role of learning in transition and the different types of learning that can benefit the users.

2.1 PERFORMANCE IMPROVEMENT WHILE USING ONE METHOD

Before discussing about the transition to more efficient methods, it is important to discuss how users' performance improves while using one method (intramodal performance [39]). It is important because concepts of intramodal improvement are going to be useful to understand and characterize the transition itself.

So this section is going to focus on theoretical and mathematical models on this kind of performance. Current theoretical models of performance improvement aim to link the users' performance with their action and the internal processes the users go through to execute the task while mathematical models aim to provide formulas that capture and quantify the users' behavior.

1 https://en.oxforddictionaries.com/definition/transition

2.1.0.1 Theories

Perhaps the most widely accepted theory of intramodal performance improvement is the one by Fitts and Posner [53]. This theory describes three stages of learning: 1. *Cognitive stage*: This is the initial stage where users try to understand how to perform the task. As a result, they are slow in their movements, and they put a lot of cognitive effort to decide which is the next step they need to do. 2.*Associative stage*: this is the second stage that users enter once they have figure out the basic steps to execute the task. In this stage, users improve faster with each trial, and they require less cognitive effort. 3. *Autonomous stage*: users enter this stage after extensive practice. In this stage performance is stable and efficient, they need minimal cognitive effort since they perform the necessary steps automatically.

Cockburn et al. [39] suggested a similar theory but focuses on computer-related tasks. They again define 3 stages: *Initial performance*, *Extended learnability* and *Ultimate performance* which correspond to the *Cognitive stage*, *Associative stage* and *Autonomous stage* respectively. The difference in Cockburn et al. work is that they emphasize the role of the system during the learning process. For example, in the first stage, the users' performance depends on how well the interface communicated the necessary steps to complete the task while for the second stage they discuss the role of interactive techniques that can help like feedback and feedforward. We are going to have a more in-depth discussion about the role of the system in performance improvement in section 2.6.

2.1.0.2 Mathematical models

Although performance can mean different things, in this thesis performance is the time required to execute a task. For example, in the case of command selection, we measure as performance the time the users need to select the command. So to improve the performance we need to minimize the execution time [2].

The most widely accepted model for intramodal performance improvement is perhaps the power law of practice [118] (eq. (2.1)). The input variable P for this model represents the amount of practice, and the output variable y is the task execution time. The model has three parameters : α is the difference between the plateau of performance and the performance of the first time the users used the method and it describes the amount of learning, β is the shape of the curve, and γ is the plateau of performance. Figure 2.1 shows the shape of the curve that the model proposes.

This model has been used to fit data of task completion times to quantify the intramodal performance. The usual procedure is that researchers collect user data from users where they have to repeat the

2 Other types of performance measurement could be for example a combination of execution time and accuracy

$$y = \alpha P^{-\beta} + \gamma \tag{2.1}$$

$$y = \alpha e^{-\beta(P-1)} + \gamma \tag{2.2}$$

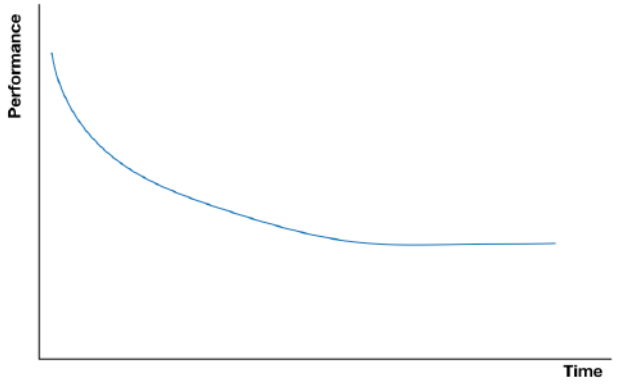


Figure 2.1: Both models can capture the performance improvment eq. (2.1)

same task for many trials. Then they average the execution time per trial, and they use some computational method to fit the model, i.e. to calculate the value parameters that can describe the data. Then researchers can use the value of the parameters to understand interesting characteristics of the task like the plateau of the performance (γ parameter).

Heathcote et al. [80] suggested that power law of practice is good for averaging data across users but not when it comes to fitting data for an individual user. They propose therefore the exponential model (eq. (2.2)) for the case of fitting data for different users. The exponential model uses the same parameters and variables like the power law of practice. The difference is the equations of each model.

Both power law of practice and exponential models offer an overview of the intramodal performance. However, they do not offer insights about the individual stages. It is interesting to investigate how we can model each stage. This is not a trivial job because it is difficult to identify where each stage starts and ends. Tenison et al. [160, 162] used hidden Markov models [132] to infer the three stages and then tried different models to fit each stage. They showed that each stage could be modeled using the power law of practice.

In summary for intramodal performance, the most commonly used model is the power law of practice (eq. (2.1)) although the exponential law (eq. (2.2)) may be more suitable when it comes to modeling data for individual users. Perhaps the most interesting point for our work is the work of Tenison et al. [160, 162]. Their work implies that it is possible to identify the three stages of performance and more importantly the

transition points between the stages. This means that there is a kind of transition even in this type of performance which we will further discuss in section 2.2.

2.2 TRANSITIONING TO A NEW METHOD

Before discussing the performance improvement while transitioning to a new method, it is important to understand the transition itself. This section first provides a set of dimensions that can describe the different types of transition as well as models of transition that describe the adoption of the new method independent of the performance.

This section focuses on the transition itself. It reviews the current literature work and aims to discuss how current theories and models characterize the transition as a phenomenon. We are going to focus on three types of characterizations: 1. characterizing by the "state" which I have already mention briefly in the introduction of the chapter, 2. characterizing by the adoption of the new method, characterizing by the performance improvement while transitioning to the new method.

2.2.1 Characterizing transition by "State"

Inspired by the terms that Cockburn et al. [39] use to differentiate the different types of performance improvement, I will define as:

- *Intramodal transition* the transition from one psychomotor state to another one while users use one method to execute a task.
- *Intermodal transition* the transition from one method to a new one to execute a task.

In this thesis, the main focus is the intermodal transition and unless stated otherwise from now on, when I refer to transition I will mean the intermodal transition.

This distinction is important since the two types have different characteristics [6]. Intramodal transition happens naturally with practice without an explicit instruction but as we are going to see in the following section intermodal transition doesn't necessarily happen with practice, but quite often users need to have an explicit instruction to switch to a new method. Additionally quite often as we discussed already there is a performance dip that users experience with the intermodal transition which is not the case when it comes to the intramodal transition. Finally it is easy to infer when users transition in the intermodal transition because we can see which method they use but in the intramodal transition we have to use computational solutions like HMM [132] (like they did in [160, 162]) or change-point algorithm [159] (like they did in [47]).

There is another aspect I want to mention here which is the case of *within-method intermodal transition*. The distinction between intra and intermodal transition implies that there is only one way to use a method, but it is possible that there are multiple ways to use the same method and some ways can be more efficient than others. Therefore it is important to emphasize that even when using one method users can experience an intermodal transition.

2.2.2 Characterizing transition by adoption of the new method

2.2.2.1 *Theories*

Current literature on adoption [39, 72, 96, 143] focuses on the duration of and the success of the transition.

The definition in the beginning of chapter described the transition as a "period" or a "process". This implies that transition can happen over time. Some of the current theories about transition though assume that transition happens instantaneously [39, 72, 143] and as result transition is reduced to a point the *switch point*. This assumption happens for simplicity reasons in order to focus on other parameters of the transition but it should be noted that as I am going to discuss in chapter 6 this type of transition can be observed in data collected from users as well.

More often than not though transition happens gradually over a period of time. Kurtenbach et al. [96] discussed that during this period the users practice the new method, but when the need they return to the old one, therefore, there is an interleaving between the two methods.

A successful transition is the one where the users actually switch to the new method.But it is not guaranteed that users will manage to switch to the new method when they start the transition. The reason for failing to transition can be explained by several reasons like lack of motivation to learn the method or the difficulty to learn the new method [72].

2.2.2.2 *Mathematical models*

To the best of our knowledge, there isn't a widely accepted mathematical model which we can use to model adoption. For the intramodal transition, we have already discussed some solutions in previous sections. Therefore, we will focus on the intermodal transition.

First, let's discuss how this model should behave [6]. This model depicts how users switch from one method to another one independent from the performance. If we assign 0 for the old method and 1 for the new method a suitable model will be the one that starts from 0 and plateaus at 1. Now based on the discussion by Anglim et al. [6] we will discuss some models and the cases they can capture.

First in the case of no transition we can use the constant model [6] (eq. (2.3)). This model takes no parameters, and it returns the same value regardless of the amount of practice. Figure 2.2 shows the shape of this model

$$y = m_{old} \tag{2.3}$$

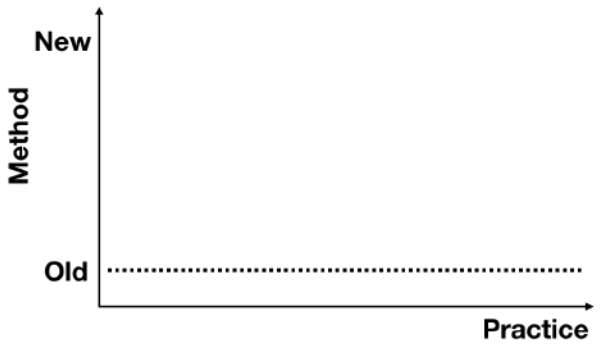


Figure 2.2: The constant model takes no parameters and it is suitable for the cases where users do not transition

Next in the case of instantaneous transition we can use the constant-constant model [6] (eq. (2.4). This model takes one parameter θ which represents the point where the users transition to the new method.

$$y = \begin{cases} m_{old} & x \leq \theta \\ m_{new} & x > \theta \end{cases} \tag{2.4}$$

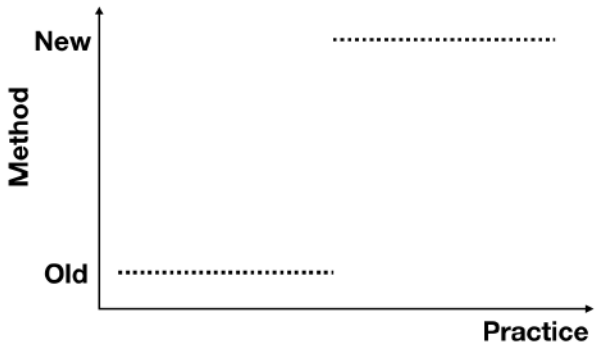


Figure 2.3: The constant-constant model takes one parameter θ which indicates the transition point from one method to the new one.

Lastly, for the case of gradual transition, all the models that start from 0 and plateaus to 1 are suitable. We are going to discuss two families of models that can be used: sigmoid functions and logarithmic functions. Sigmoid functions have been used to model psychophysical data [176, 177] where they try to see how users' behavior evolves when the intensity of a given stimulus increases. We will discuss here the logistic function [9] but other functions like Gumbel and Weibull functions [164] (eq. (2.5)) can also be used. The logistic model takes 3 parameters: α which represents the upper asymptote of the curve, μ which represents the inflection point aka the point that the curve reaches the 50% of α and σ which represents how quickly the curve raises to α.

$$y = \frac{\alpha}{1+e^{-\frac{(P-\mu)}{\sigma}}} \tag{2.5}$$

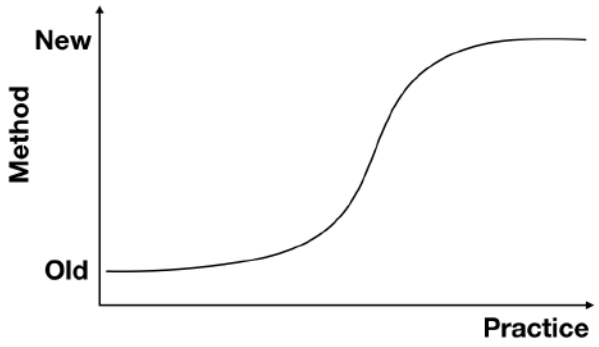


Figure 2.4: Sigmoid functions can model the gradual transition. They usually have three parameters that can provide useful insights: α which is the upper asymptote , μ which is the inflection point i.e. the 50% of *alpha* and σ the speed of the raise to α

The second family is the logarithmic functions [29]. This family of functions starts from 0 and raises until it reaches a plateau. Figure 2.5 shows an example of the form of these functions. We are going to talk about two of these functions. The saturating exponential function [29] (eq. (2.6)) takes three parameters where γ is the method used in the first trial, $\gamma+\mu$ is the asymptote of the curve and σ shows how fast the curve raises. Michael-Menten [29] (eq. (2.7)) also takes three parameters with σ indicating how fast the curve raises, μ the inflection point and γ is the method used in the first trial.

$$y = \gamma(1 - e^{-\sigma(P-1)}) + \mu \tag{2.6}$$

$$y = \frac{\sigma(x-1)}{\mu+(P-1)} + \gamma \tag{2.7}$$

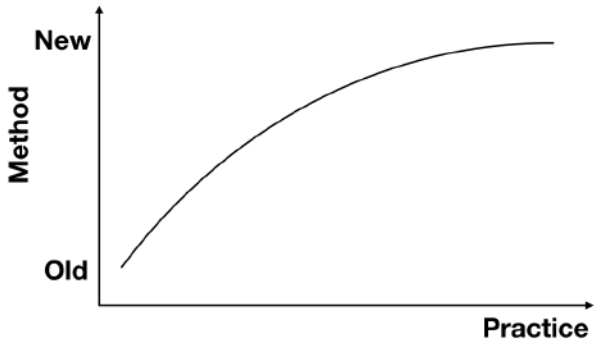


Figure 2.5: The saturating exponential function models also the gradual transition

There are several points that I want to discuss about these models. First, none of the discussed models capture the case of failed transition [3]. In my opinion, a suitable model to capture this case is a model that starts from 0 raises until a point and then returns to 0 again forming essentially a bell curve. Second, to the best of my knowledge there hasn't been any work on whether some of the above models are better for modeling grouped data or individual users' data like in the case of eq. (2.1) and eq. (2.2). Third, although both sigmoid and logarithmic functions capture the the asymptotic amount of the new methods' adoption and how fast the curve reaches this point, the shape of the sigmoid captures the initial period where users use the old method. Finally the above models focus on the method adoption and not the users' performance while using transitioning to the new method. The next section focuses will elaborate more on this performance improvement.

2.2.3 Characterizing transition by performance improvement

How performance improves while users transition to a new method is an important topic. I am going to refer to this performance as inter-

3 By failed transition I mean the case were the users start the transition to a new method but after some practice they decide to return to the old one

modal performance which is the term that Cockburn et al. [39] used in their own framework.

2.2.3.1 Theories

Current theories [39, 72, 143] of intermodal improvement make two assumptions. First, they assume that users have already reached the plateau of performance with one method and second that transition happens instantaneously. Both assumptions are made for simplification reasons, but as I discussed before transition can happen over a period of time. Nevertheless, these theories can still offer us insights about the intermodal performance and the users' behavior. Another assumption is that when transitioning to a new method users experience a *performance dip* with the new method until they reach a point of a *performance gain* where the performance of the new method is better than the old one. So current theories link users' behavior with some or all of the following points of interest: 1. the plateau of performance with the previous method, 2. the point where they transition to a new method, 3. the performance dip and 4. the performance gain.

Gray et al. [72] proposed the Plateaus, Dips and Leagues theory to explain how users discover and transition new methods while performing the task. They have identified three stages: 1. *Method Invention, Discovery, or Instruction*: this is the period where the users start to wonder if there is another method to perform the task. They identify this stage as the period where the users have reached the plateau of performance with the previous method. 2. *Method development*: this is the stage where the users begin to learn the new method, and it is possible that they are going to experience a performance dip. 3. *Practice*: this is the stage where the users have started to get better with the new method, and they already see a performance gain. They haven't necessarily reached the plateau of performance with the new method yet, but with practice, they will eventually do.

Cockburn et al. [39, 143] also discussed the intermodal performance in their work. They also assume that transition happens after reaching the plateau of performance with the new method and that it happens instantaneously. They focus on the transition point, and the performance dip and they discuss factors that affect those two points. We are going to discuss these factors in section 2.4. An interesting point that they raise is that the performance dip can prevent users from making the transition.

2.2.3.2 Mathematical models

Modeling the intermodal performance improvement is not an easy task, and as a result, there aren't widely accepted models as in the case of intramodal performance improvement. One model that has been used in some studies (e.g. [46, 160, 162, 182] is the piecewise learning curve.

This model consists of multiple learning curves. The appropriate curve is chosen based on the amount of practice the users have.

$$y = \begin{cases} \alpha_1 P^{-\beta_1} + \gamma_1 & x \leq \theta \\ \alpha_2 (P - \theta)^{-\beta_2} + \gamma_2 & x > \theta \end{cases} \tag{2.8}$$

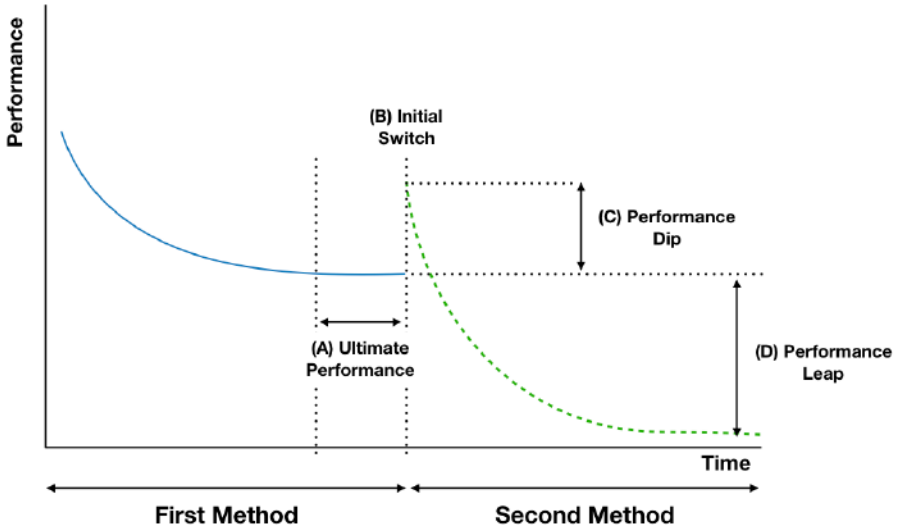


Figure 2.6: Intermodal performance improvement [39]

Equation (2.8) shows an example of the piecewise power law equation. In this equation y is the task completion time , P is the amount of practice, α_1, β_1 and γ_1 are the parameters for the first learning curve, α_2, β_2 and γ_2 are the parameters for the first learning curve and θ is the transition point from one method to the new one. From this model we can find: 1. the plateau of performance with the previous method from γ_1, 2. the transition point from θ, 3. the performance dip from difference between the $y(\theta) - y(\theta + 1)$ and 4. the performance gain from the difference $\gamma_2 - \gamma 1$.

It should be noted that other variations of the above equation can be possible. For example we can use the exponential law (eq. (2.2)) instead of the power law of practice (eq. (2.1)). Also, we can perhaps indicate prior experience with the new method by subtracting another constant in the second part of the equation instead of θ.

2.2.4 Discussion

To summarize, this section discusses theories and models about the performance while using one method and while transitioning to a new one. Although those theories have some shortcomings to be more simple, we can get some interesting information about the transition. Perhaps the most interesting insight is that there is a transition on the

intramodal level and the intermodal level. The main shortcoming of these theories in our opinion is that they view the transition as something that happens immediately while as we are going to discuss in the next section transition can occur over a period of time.

2.3 FAVORING SUBOPTIMAL METHODS

In the previous sections we talked about the transition to more efficient methods, but studies suggest that users actually fail to do this transition. Bhavnani et al. [25, 26] found that even users with many years of experience still preferred suboptimal methods. In this section, we discuss theories and factors that can explain users behavior.

Many theories have been proposed to explain why users do not switch to more efficient methods. The most prominent one is the "active user paradox" [32] which posits that users tend to prefer short-term productivity to long-term effects due to a productivity and assimilation bias [32]. The production bias refers to the users' tendency to finish their task quickly without caring spending time to learn alternative methods. The assimilation bias refers to the users tend to prefer the methods they already know even if they suspect there might be more efficient ones. Einstellung effect [105] supports the same behavior as the assimilation bias. Simon et al. [149] also noted that users choose the method that is going to satisfy their goals quicker even if there are better ones. Anglim et al. [6] discussed that another possible explanation is the escalation of commitment bias [154] which posits that users tend to stick with their choices even if they feel that it has a negative effect. This implies that users may still choose suboptimal methods even if they feel that the cost of choosing them is too high regarding performance. Fu et al. [58] found that users prefer methods that provide interactive feedback at each step. This is especially relevant to the command selection task because efficient methods like keyboard shortcuts do not provide such feedback. Gray et al. [69–71, 73] proposed the soft-constraint hypothesis to explain why the users prefer suboptimal methods. They posit that users consider interactions that give quick feedback (0.3 seconds up to 3 seconds) as efficient even though there might be better alternatives.

From the above theories, we can already see some common factors that users take into account when it comes to choosing which method to use:

Practice: Most of these theories point out that users tend to choose methods that they have previous experience e.g. Einstellung effect [105], the assimilation bias [32] and also empirical data from Fu et al. [58] also support this observation. This is an interesting observation since the theories about intermodal improvement assume that users first reach

the plateau of performance with the previous method to switch to the new one, while here we notice that experience may be an inhibitory factor.

Interactive feedback: Users also tend to prefer methods that are interactive and provide incremental feedback [58]. This could be explained by the production bias [32] which posits that users prefer methods that produce quick results.

Effort: Studies (e.g., [56, 57]) based on the rational analysis framework proposed by Anderson [5] suggest that users tend to prefer methods that require the least effort.

2.4 FACTORS FOR PROMOTING EFFICIENT METHODS

The question is then how users transition to more efficient methods. In Cockburn et al. [39, 143] work on intermodal performance they discuss factors that help users make the initial switch to the new method and to minimize the performance dip. It should be emphasized that the factors from the previous section can also help users transition to a new method given that they apply to it. The following factors are interesting because interaction techniques uses them to promote shortcuts (see chapter 3).

2.4.1 Initial Switch to a new method

This section discusses factors that can help users make the initial switch to the new method.

Awareness of the new method: One of the most important factors for a user to switch to a new method is to be aware of it.

Perception of future performance : For users to consider switching to a new method they have to believe that the new method is going to help them complete their tasks [172] and that the new method is more efficient than the old one [39].

Motivation: Lack of motivation can hinder the adoption of a new method. This concerns both intrinsic (i.e., users' own willingness to improve themselves) and extrinsic motivation (factors from their environment like time pressure [158] motivate the users to improve).

The above factors can be originated by the users (e.g. intrinsic motivation) but also their environment can play an important role. What is environment? For simplicity reasons, I am going to make a distinction between two environments. The first is the interface environment which the users work on. The question here is how designers can create an environment that can help users discover more efficient methods. This is on of my main research questions on this thesis and following sections are going to discuss about this.

The other environment is the social environment of the user. An example of this can be the working space of the users. The question here is how the state of this environment (e.g. social interaction with other users, time pressure etc) can affect users behavior. Although this type of environments are not the focus of this thesis, they are nevertheless interesting and worth exploring. For example existing studies [122, 123] indicate that the users' social environment can indeed affect their behavior yet other studies [128] indicate the opposite.

2.4.2 Performance Dip

The magnitude of the performance dip can potentially hinder the adoption of the new method, and as a result, users will return to the previous method. Helping the users to learn the new method faster can help to reduce this dip.

Learnability of the new method: Another factor that can affect the performance dip is how easy to learn the new method. Of course, each method has its characteristics that can affect the learning process but there are some basic insights can accelerate this process. Providing feedback and feedforward [42] can help users learn the new method. It should be noted though that according to the guidance hypothesis [144] too much help can negatively impact the learning process. Additionally, Kurtenbach et al. [95] suggested that physical rehearsal can help users learn the method faster.

2.5 THE ROLE OF LEARNING IN TRANSITION

The previous section discusses the importance of how easy is to learn the new method in transitioning to it. In high level, learning can be defined as "a behavioral change that results from experience" [165]. This is a very broad definition, and there are many types of learning discussed in current literature[4]. In this section, we are going to discuss two types of learning which differ by whether the users learn consciously (explicit learning) or unconsciously (implicit statistical learning). I focus on these types of learning because some of the existing interaction techniques are based on these learning types (e.g. [77]) and also part of the work in this thesis is based on them as well (chapters 4 and 5).

2.5.1 Explicit learning

Explicit learning is defined as "learning which generates verbal knowledge of movement performance (e.g., facts and rules), involves cog-

4 for example incidental, intentional and associative learning

nitive stages within the learning process and is dependent on working memory involvement" [91]. This type of learning follows a more traditional approach where the users go through all the stages of the intramodal improvement starting from the cognitive stage until they reach the autonomous stage. An interesting implication for this type of learning is that users first have to be aware of the method before starting learning it.

2.5.2 Implicit Statistical Learning

Two types of learning are associated with unconscious learning. First, the implicit learning which is defined as "learning which progresses with no or minimal increase in verbal knowledge of movement performance (e.g., facts and rules) and without awareness. Implicitly learned skills are (unconsciously) retrieved from implicit memory" [91]. Second, statistical learning [140] refers to an unconscious cognitive process in which repeated patterns, or statistical regularities (e.g., probabilistic regularities of the environment that predict future events), are extracted from sensory inputs [168].

These two types of learning seem similar. Some publications consider that they are synonymous since the studies from the two types of learning produce similar results (for an overview see [124]) to the extent that Conway et al. [41] coined the term Implicit Statistical Learning (ISL). Implicit and statistical learning differ on a conceptual level on how they interpret the internal learning processes [124] so in their essence they describe the same phenomenon but offer a different interpretation of why it is happening. For the rest of this thesis I am going to adopt Conway et al. approach and refer to them with the term of ISL.

Finally ISL is related to chunking [64, 125]. The term chunking is used to characterize the associative processing by which people bind together co-occurring elements or information from their interaction with the environment. According to the main influent chunking models, attention plays a critical role in the formation of cognitive units: perceptual primitives would only be grouped together to form a chunk when they are simultaneously held in a spatial-attention window, which is constrained by working-memory limitations [126].

Current research of ISL has yielded some interesting results. First findings have shown that ISL is involved in may tasks including sequence learning (e.g., [**haith1993fornmation**]), visual search (e.g., [35, 66–68, 163], attentional guidance[20, 166], word-object association (e.g., [151]), cue-category association (e.g., [141]), causal learning [65] and contextual cueing [68] which are relevant in HCI [171]. Second, ISL can lead to faster automatization usage of the method [87] without having to go through the cognitive stage of intramodal improvement. Finally, ISL can occur while users do a different task [142, 169, 170], meaning that users can implicitly learn a method to execute a task while doing

something else in the interface which is particularly interesting in the command selection task.

2.6 IMPLICATIONS FOR SYSTEM DESIGN

Previous sections focused on users behavior but one of this thesis' goals is how to design systems that promote more efficient methods. So how the previous discussion can inform the design of interactive systems? Based on section 2.4 I argue that a system should 1. make aware to the users of existence of the more efficient method, 2. inform users about the benefits of using this method and 3. help users learn the new method. The design decisions that designers have to make in order to have a system with the above three characteristics depends on the task but still there are some tools that can be used in most if not all tasks.

2.6.1 Feedback and Feedforward

Schmidt et al. [145] discussed three types of information that can help the users' performance while using a method. The first type is the information the users have before doing the task, the second type is the information that users receive while doing the task and the third type is the information the users receive after they completed the task. These types of information are related to interaction design principles like feedback and feedforward.

Indeed part of current literature has focused on investigating the different types of feedback and feedforward and their importance on user experience (for an overview see [39, 49, 145, 173]). In addition as I am going to discuss in the next chapter many interaction techniques regarding command selection use those two principles (e.g. [96, 107]) to promote Keyboard shortcut (KS).

2.6.2 Contextual cueing

Feedback and feedforward offer a direct way to help the user transition to a new method and learn it. But as I discussed in section 2.5 designers can take advantage on Implicit Statistical Learning (ISL) also. ISL principles can benefit the design of an interface because users can potentially start learning the new method while using the old one [142, 169, 170].

I am going to focus on a particular form of ISL called contextual cueing. It illustrates how during visual search, contextual regularities present in the display can be implicitly detected and learnt, allowing an optimization of basic visual processing and/or attentional deployment in subsequent encounters (for a review, see [68]).

In its standard experimental task [163] in cognitive science/psychology, participants search for a T-target within a configuration of L-distractors. Half configurations are systematically repeated across many blocks of trials. The others are presented only once during the task. A benefit on search times is typically observed in the repeated contexts compared to the novel contexts. This indicates that participants *implicitly* encode some elements of the context, without being aware of that.

So contextual cueing can be interesting when it comes to designing a Graphical User Interface (GUI). A GUI is composed of various graphical elements that inform the users about the capabilities of the system which is especially true in the case of command selection. Therefore, how and where these elements are positioned can affect the users' behavior.

Contextual cueing appears sensitive to several factors ([61], for a review, see [68]), such as:

Spatial proximity. Spatial proximity is one of the most important factors of contextual cueing, which is directly related to the distance between two elements. Although incidental learning on non-spatially closed elements is possible, it occurs under far more restrictive conditions than those required for learning the relations between spatially close events [68, 125]. This phenomenon is related to the *law of proximity* by the Gestalt Theory, in which to perceive an assortment of objects, an individual perceives objects that are close to each other, forming as a group. Note that this law is often used in advertising logos to emphasize which aspects of events are associated. For example if designers move closer the KS to where the users main focus on attention is (e.g. the name of the command they are looking for) they can potentially be motivated to use them.

Temporal proximity. A delay of 3000ms between two statistically contingent elements was also sufficient to deteriorate intertrial-learning [163] in a contextual cueing task, indicating that the timing of users' exposure to an element shortcut in learning a new method (for example if a user is exposed often to a keyboard shortcut then it is possible to learn it).

Accumulation. Another factor can be seen in the accumulation of instances in memory. Indeed, research in SL shows that only limited contingencies can be learnt in a restricted period, suggesting that only a few associations trigger learning [152]. In the context of menu to KS transition for example, this suggests that limiting the total number of KS cues [5] can facilitate the implicit learning of the remaining ones.

5 by cue I mean the GUI element that represents the keyboard shortcut

2.7 CONCLUSION

To summarize this chapter covers related work on transition to expert methods and learning processes. Although we have discussed various issues, we want to emphasize three points that are relevant to the work of the current thesis.

First theories about the intermodal performance don't take into account that transition usually happens over a period of time. As a result we don't have enough insights about the users' behavior during this period which could potentially help us understand how to promote expert methods.

Second there are different possible models focusing on the transition to a new method but we do not know how well they work in practice. Models can be a useful tool to analyze and quantify users' behavior therefore being able to use these models in practice can actually be very useful.

Lastly theories of learning raise some interesting directions for Graphical User Interface (GUI) design. Part of the work in chapters 4 and 5 will explore how spatial proximity can potentially benefit the users.

We are going to revisit all three points during this thesis. In the next chapter we are going to focus on command selection methods so we will discuss more about system design and statistical- explicit learning.

3 COMMAND SELECTION METHODS : DESIGN AND INTERACTION TECHNIQUES

In this chapter, we define, describe and discuss the main characteristics of methods for selecting commands in GUIs. I first distinguish RECOGNITION VS. RECALL based methods. I then describe a framework highlighting the three main elements of a command and the semantic and graphical relations between these elements. In particular, I show that the analysis of these relations offers opportunities for design that we investigate in the following chapters. Finally, I discuss interaction techniques favoring keyboard shortcuts

3.1 CURRENT COMMAND SELECTION METHODS

We can categorize the current command selection methods into two groups based on where the users have to look primarily for the necessary information to use the method. The first category is RECOGNITION-BASED methods which places the information (knowledge) in the Graphical User Interface (GUI). This means that users need to look into the interface and interpret the displayed information to select the command they want. The second category is RECALL-BASED methods which places the information in the head of the users. This means that they need to recall the necessary steps to use the command from their memory.

3.1.1 RECOGNITION-BASED methods

The most common RECOGNITION-BASED methods that users encounter in current GUI are linear menus, toolbars and ribbons.

Linear menus (fig. 3.1 - A) are hierarchical lists that can contain either commands or other menus. Usually, the users can find the menus in a dedicated area on the top of the window called menubar. Menubar contains multiple menus, and each of these menus has a specific title. To open the menu, users have to click on the menu's title. Apart from the menubars, linear menus are used when users press the right mouse button. These are called context menus.

Toolbars (fig. 3.1 - B) are graphical widgets which contains a list of buttons each of them associated with a command. Usually, the buttons include a pictorial representation of the command which helps the users recognize which command a given button represents. Toolbars contain a subset of the available commands, usually the ones that are used more frequently.

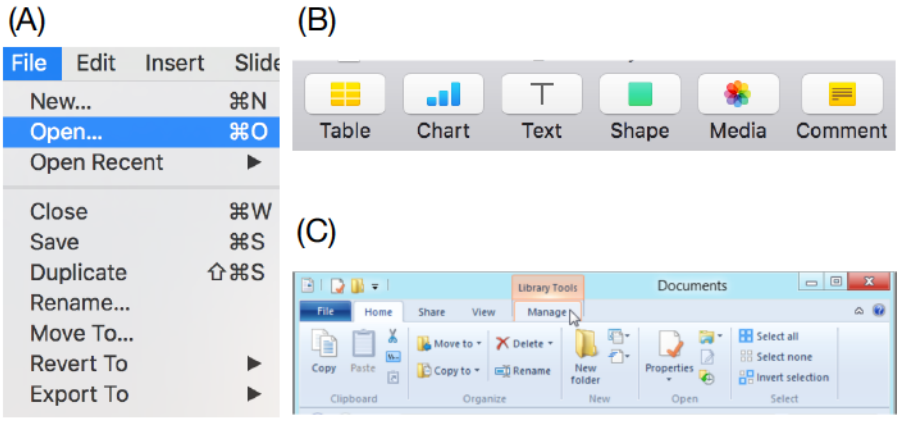


Figure 3.1: Recognition-Based methods: (A) Linear menu, (B) Toolbar, (C) Ribbons, (D) Keyboard shortcuts, (E) Command line interface and (F) Stroke shortcuts.

Ribbons (fig. 3.1 - C) are a special case of toolbars. Ribbons organize the available commands into categories and each category contains a toolbar with the available commands of this category [111].

It should be noted that as the amount of the commands grow, the task to organize the commands in a more meaningful way becomes more challenging. As a result, we can see now applications where they combine together some of the above methods. For example, in addition to icons ribbons can have hierarchical menus or other widgets like radio buttons and checkboxes. Additionally although this thesis focuses primarily in linear menus there are other menu layouts in current literature like the circular menu [139] and grid menus [1].

3.1.1.1 *Method Characteristics*

These methods share some key characteristics that Bailly et al. [13] have identified.

They allow users to select commands from a set of items [54]. The difference is that ribbons and linear menus could potentially include all the available commands while toolbars usually contain a subset of the available commands.

They have a structure to display the items visually. In most applications, they can contain multiple level hierarchical groups, and they can even group items on the same level using separators.

Finally, linear menus are transient [86]. This means that they do not occupy permanent screen space. They appear when the users open them and disappear when the users close them. Toolbars and ribbons, however are always visible and occupy a predefined screen space.

3.1.1.2 *Design Guidelines*

The effectiveness of the these methods depends on how easy it is for the users to interpret the information displayed. So designers have to create structures that include a lot of commands that users can easily navigate.

To that end, both Apple and Microsoft [8, 113] have provided similar guidelines. For example they suggest that designers should use meaningful names for the commands and the menu titles. The methods should not contain many sub-levels and they should not contain many items. Designers should use verbs for commands that initiate actions and adjectives for commands that modify the attributes of an object. Finally the menu items should be easily accessible and in the case of linear menus the commands should be on the top.

Another important thing that designers should consider are the icons. Icons should successfully communicate the meaning of the command. This means that an icon should be semantically related to the meaning of the command, and it should be comprehensible and distinct from the other icons. Current literature [104, 106, 109, 115] aims at characterizing the similarity between the graphical representation of an icon and the meaning of the associated command. For instance, Lodding [104] distinguishes three types of semantic relations: representational, abstract and arbitrary. Representational icons rely on typical and intuitive objects to represent the command meaning. Abstract icons are composed of geometric shapes, whereas arbitrary icons do not have intuitive connection between the icon and the meaning of the command. Finally an icon should also have simple design and be aesthetically pleasing [63, 104, 156]. Its elements should be orderly grouped and to have distinct figures [102].

3.1.2 Recall-Based methods

Figure 3.2: Recall-Based methods methods: (A) Keyboard shortcuts, (C) Command line interface and (B) Stroke shortcuts.

The most common methods in this group that users encounter in current GUI are the Keyboard shortcut (KS), Stroke Shortcut (SS) and Command Line Interface (CLI).

KS (fig. 3.2 - A) are keys or a sequence of keys that when the users press to select command. They typically contain a series of modifiers

(e.g. Ctrl ,Alt ,Shift for Windows and control ,⌥ ,⇧ and ⌘ for MacOS) followed by a letter like Ctrl+C for invoking the command "Copy" or ⇧+⌘+Z for invoking the command "Redo". There are also single letter keyboard shortcuts , i.e. without any modifiers like P for selecting the pencil in mouse based applications like Adobe Photoshop.

A special category of KS which is noteworthy are the access keys [178] . Access keys are used to make command selection with the keyboard more accessible. When the linear menu is open if the users press the Alt key then a letter of the command name is underlined. If the use press this letter then they can select the command. See fig. 3.3 for an example.

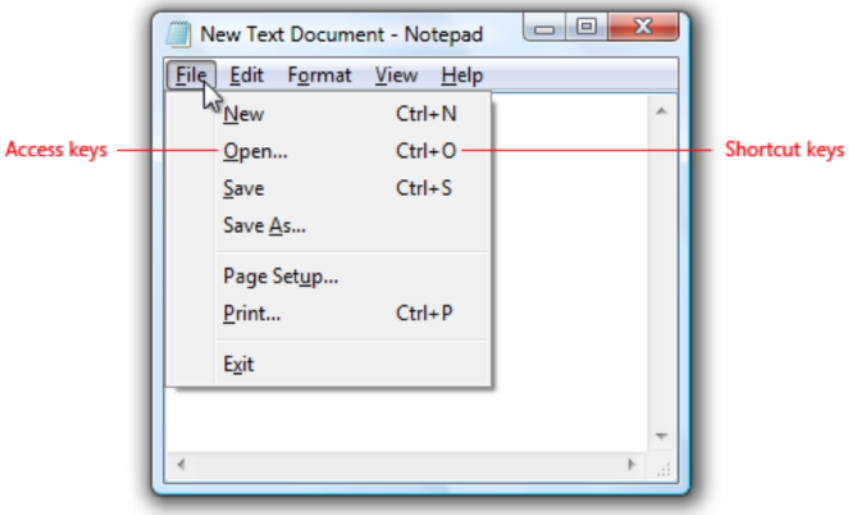


Figure 3.3: An example of access keys [178].

SS (fig. 3.2 - B) stroke shortcuts are gestures that represent the movement trajectory of an input device, e.g., a mouse or a pen or the user's finger, e.g., when using a touch-pad [184]. The users make these trajectories to draw a symbol which invokes the command.

CLI (fig. 3.2 - C) allows users to select commands using a prompt. The users enter commands in the prompt in the form of text lines (one line per command).

3.1.2.1 Method Characteristics

Keyboard and stroke shortcuts share very similar characteristics. They both allow users to select a subset of the available commands. They usually appear as a label (iconic or textual or both) right of the command in the menu or as tooltip if the users hover over an icon. Thus they do not require a lot of screen real estate. Their main advantage is that they provide a direct way to select commands by minimizing the interruption of their current task [13, 107].

These methods have two main differences. First, KS uses the keyboard as the input device while SS uses a pointing device like a mouse. Also, SS could potentially be easier to remember than KS because they are iconic [7, 184].

Finally I want to discuss where users can find information about these methods (e.g. the keyboard shortcut combination to invoke the command) in current interfaces. KS is perhaps the most commonly used of the three methods and as a result usually RECOGNITION-BASED methods are responsible to display the KS combination (see section 3.2). This is not the case for the other two methods though. Both methods don't have a standard way to appear in current GUI. CLI methods don't appear in the GUI at all. Users use a command prompt that they can access through an external terminal application or like in the case of atom[1] through a terminal that the application itself offers. SS don't have a commonly accepted way to appear in current GUI. There are some cases thought that they appeared where the KS usually appears ([7, 189]. Since one of the goals of this thesis is how to revisit current layout practices to better communicate more optimal methods for the rest of the thesis I won't discuss CLI.

3.1.2.2 Design Guidlines

The main design guideline for both KS and SS is that they should be easy to learn so that users can use them without much effort.

In the case of KS the challenge is to carefully choose the modifiers and the letter. Modifiers should be chosen based on the meaning of the command. Design guidelines from both Apple and Microsoft [88, 112] encourage to use the control and Ctrl as the main modifier. Designers should use ⇧ and Shift to extend or compliment existing KS. For example if Ctrl+Z invokes the Undo command then Shift+Ctrl+Z would invoke the "Redo" command. Finally these guideline suggest to use ⌥ and Alt sparingly. Finally designers should consider letters that are associated with the command. A common practice for example is to use the first letter of the command as KS letter (e.g. Save - Ctrl+S).

In the case of SS, designers should consider the complexity of the gestures. Zhai et al. [184] discussed three orders of complexity. Zero-order gestures like a soft tap on the touchpad. First order stroke are unistroke gestures. Higher order are gestures that are being drawn sequentially or parallel. Additionally as with KS users should be able to associate the stroke with the command it represents.

3.1.3 Command selection methods and user behavior

What are the benefits of these methods? In this section, I present and discuss the findings of several empirical studies comparing these meth-

1 atom.io

ods. The findings are organized according to 4 criteria: (1) *intramodal performance improvement*, (2) *forgetting and relearning* a method, (3) *method adoption* and (4) *intermodal performance improvement* .

3.1.3.1 Intramodal performance improvement

According to Norman et al. ([119] the two group of methods differ in terms of intramodal improvement. Recognition-Based methods tend to be easy to use from the beginning and because they rely on interpretation of the displayed information they are easy to learn. In contrast, users need more time to use the Recall-Based methods methods since the users need more time to learn them (e.g. a user need to learn the keyboard shortcut combination). In the end though the Recall-Based methods are faster than then the Recognition-Based methods. In terms of Cockburn et al. framework ([39]) this means that Recognition-Based methods are going to be faster in the *Initial Performance* stage and perhaps partially in the *Extended learnability* stage but in the *Ultimate Performance* stage the Recall-Based methods outperform the Recognition-Based methods.

Results from empirical studies (e.g. [120, 133] that compare the two groups (usually they focus on menus, toolbars and keyboard shortcuts) seem to confirm Norman's observations. These studies usually assign for each user a method and they ask them to select commands using one of these methods. They compare the average performance per block of trials or the overall performance in order to see which method tends to be faster. Results indicate that Recognition-Based methods tend to be faster than the Recall-Based methods for the initial blocks ($\approx$ 15 blocks) but in terms of ultimate performance Recall-Based methods outperform the Recognition-Based methods (e.g. [89, 120])

3.1.3.2 Forgetting and Relearning

One aspect that I did not discuss in the intramodal improvement section is the performance when users need to relearn a method. We can expect that after a period of not using a method, users might forget how to use it and therefore they might experience a performance dip when they try to use it again. So it is interesting to see how the two categories are affected by this period.

Given that Recognition-Based methods rely in recognition while Recall-Based methods rely on recall, an initial assumption is that users will be require more time to relearn them than the Recognition-Based methods. Empirical studies also point to this observation. Kim et al. [89] compared the performance of linear menus and KS and they observed that users tend to be faster when users try to relearn linear menus than KS. Remington et al. [133] noted that the relearning performance dip can be decreased if the method is easy to recall. For example if the

combination of a KS is easy to remember then users might have easier time to recall it even after a period of time that they haven't used it.

3.1.3.3 Method adoption

Field surveys [99, 122, 123] indicate that computer users even with many years of experience prefer using toolbars and linear menus over keyboard shortcuts. They only use keyboard shortcuts for very common commands like Copy and Paste.

In addition several studies have tried to measure the transition from Recognition-Based methods to Recall-Based methods. Many of these studies (e.g. [77, 107, 189]) compare interaction techniques to promote more efficient methods. Therefore they usually have a baseline condition where the users see the traditional interface and then they have other conditions where they can use the interaction technique. To describe the adoption to more efficient methods (in this case the Recall-Based methods) they calculate the amount of times users used these methods over a block of trials .In the baseline condition results usually indicate that users prefer the Recognition-Based methods.

Finally an interesting finding regarding the method adoption comes from a study by Appert et al. [7]. In their study they compared linear menus, KS and SS. One of their findings is that users preferred SS over KS and they performed better with them. This can be explained because linear menus and SS use the mouse as an input device, therefore they are more similar than linear menus and KS (KS use the keyboard as an input device).

3.1.3.4 Intermodal performance improvement

Current studies about command selection method comparison do not focus on measuring intermodal improvement. In the previous chapter I discussed that measuring this kind of performance can be difficult and that there aren't any obvious ways to do it. One possible obstacle is to identify the periods where the dip and leap of performance happens which is not straightforward.

3.1.3.5 Discussion

Based on empirical observations and theoretical frameworks we can see some interesting differences between the 2 groups. Ultimately Recall-Based methods tend to be more efficient than Recognition-Based methods but they require a longer learning period and users can potentially forget them if they do not use them. Finally users tend to prefer Recognition-Based methods over the Recall-Based ones.

It should be noted though that current studies that compare these methods do not usually use models or other tools that could provide better insights. For example using the one of the learning curves (eq. (2.1)

or eq. (2.2)) can potentially give better insights about the intramodal performance. The same applies for the adoption of new method. Using one of the models (e.g. eq. (2.5)) could give better insights about the adoption rate. Finally there is little focus on measuring the intermodal improvement which is expected since from what we saw from the previous chapter there aren't many tools to properly measure and model it.

3.2 INTERFACE DESIGN PRACTICES FOR COMMAND SELECTION METHODS

In this section, I discuss how designers implement the different methods, i.e. how do they choose the different elements (command name, command icon, keyboard shortcut) of a command (semantic level) and how to represent them in a GUI (graphical level)>)[2]

Glossary:
***command name** : the name of the command i.e. the system functionality.*
***command label** : the graphical representation of the command name .*
***command icon**:the icon that represents the command.*
***Keyboard shortcut** (KS): the key combination that invokes the command.*
***hotkey(s)**: the letter(s) of the KS key combination.*
***KS cue** : the graphical representation of the KS.*

3.2.1 Design practices

3.2.1.1 Practices for designing each element

Although design process may differ for each designer, I make the assumption that there are two phases : 1. the semantic design and 2. the graphical design.

By semantic design I refer to the phase where designers decide the meaning of each element i.e. what kind of information the element intends to communicate to the users [28, 76, 180]. To that end there has been a body of work that has focused on lexical semantics (e.g. [136, 147]) which refers to the semantics of a word and on pictorial semantics (e.g. [40, 183] which refers to the semantics of an icon.

One aspect of semantics that highlights the relationship between two elements is *suggestiveness* [135, 136]. *Suggestiveness* refers the mapping of the semantics between two elements i.e how the semantics of one element refers to the semantics of a another. The most straightforward example is how a word such as "Save" or an icon like refers to the functionality of the system that focuses on saving a file.

Graphical design deals with all the decisions regarding the appearance of the command elements. For example which shapes will the command icon include and which font will the command name have.

It should be noted that there are probably more aspects involved in this process, and even these two phases can be broken to sub-phases. The reason I focus to what I call semantic and the graphical phase will become more apparent in section 3.2.2 where I will talk about the relationship between the elements.

2 I will not focus on Stroke Shortcut (SS) because they do not appear in traditional interfaces.

COMMAND NAME

A command name should provide a meaningful description of the system functionality that it represents. Design guidelines [8, 113] suggest to use verbs for commands that initiate actions and adjectives for commands that modify the attributes of an object.

Graphically, a command name appears in the interface as textual label (command label). The graphical properties of a textual label that the designers have to deal with are font size, font family, color etc. Although the font family and size may differ, usually command labels have black color to indicate that the command is selectable and gray to indicate that the command is not selectable.

COMMAND ICON

Command icons should successfully communicate the system functionality. Current literature [104, 106, 109, 115] aims at characterizing the similarity between the graphical representation of a command icon and the meaning of the associated command. For instance, Lodding [104] distinguishes three types of semantic relations: representational, abstract and arbitrary. Representational icons rely on typical and intuitive objects to represent the command meaning. Abstract icons are composed of geometric shapes, whereas arbitrary icons do not have intuitive connection between the icon and the meaning of the command.

Graphically, a command icon should also have simple design and be aesthetically pleasing [63, 104, 156]. Furthermore it should be comprehensible and distinct from the other icons.

KEYBOARD SHORTCUT

A KS should have a key combination which is easy to learn so that users can use it without much effort. Designers choose both the modifiers and the letter carefully. For the modifiers design guidelines [88, 112] encourage designers to use he ⌘ (for Mac OS) and Ctrl (for Windows OS) as the main modifier. They should use ⇧ and Shift to extend or compliment existing KS. For example if Ctrl+Z invokes the Undo command then Shift+Ctrl+Z would invoke the "Redo" command. Finally they should use ⌥ and Alt sparingly.

Graphically, a KS also appears as a textual label (KS cue) in the interface. Usually the KS cue shares the same font properties with the command label. Depending on the operating system KS cue may represent the modifiers with symbols in their names for example the *Shift* modifier can appear like Shift in a Windows O.S. and like ⇧ in a Mac O.S.

3.2.1.2 *Practices for layout design (interface design)*

I now discuss how the three elements (command name, command icon and keyboard shortcut) are organized in linear menus and toolbars.

I reviewed menus and toolbars in different operating systems (Windows, Mac and Linux), the design guidelines that these operating systems offer [8, 111–113] as well as existing toolkits (e.g. Qt, Swing). Even among different systems, we observed that the layout remains more or less the same.

LINEAR MENU

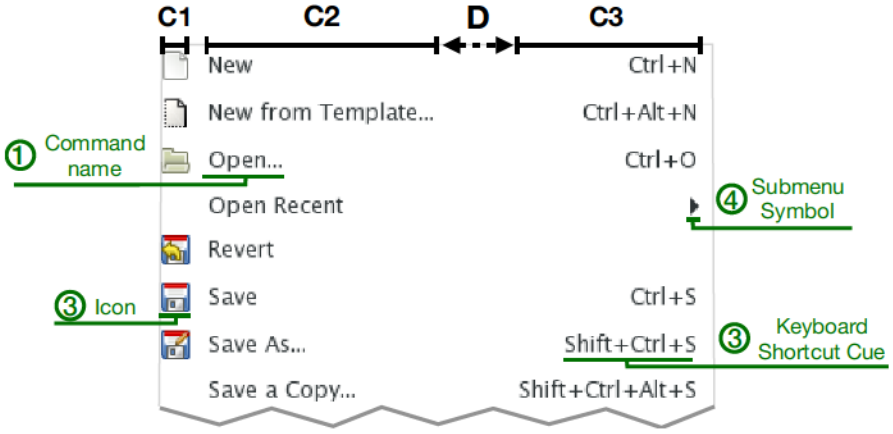


Figure 3.4: An example of a convectional menu layout of the inkscape application.

A linear menu (example in fig. 3.4) usually organizes the elements in three columns:

- **First column**: The first column contains command icons. Such icons depend on the application and the operating system, e.g., most Mac menus (as well as some Linux version) do not display icons by default. In some cases, a symbol can replace an icon like ✓ or a widget like a toggle button ⊙ or a check box ☑. In most menus, the command icons are only shown for the frequent commands. The *alignment* of the icons is fixed as they always have the same size.

- **Second column**: The second column contains command labels. Ellipses can be added [84] (e.g. Save As...), if the command requires parameters (i.e. the activation of command opens a dialog box). We also observed some instances (e.g. history menu in Safari) where the command labels use an icon (e.g. the icon of the website). The *alignment* of the command names is always on the left.

- **Third column**: The third column contains *submenu symbols* (►) and *keyboard shortcut cues*. Depending on the operating system the label can be textual (e.g. Ctrl+S) or, as in Mac systems, a combination of symbol and text (⌘S). The symbol ► is used for

hierarchical menu items to indicate that the given item opens a sub-menu. The sub-menu items do not have a shortcut cue. The *alignment* of the keyboard shortcut cues is always on the right.

We found few exceptions of the above standards. For instance, KS cues were left-aligned in an old Window version (Win 3.1). In addition some menus used an additional (fourth) column (in Windows) dedicated only to the submenu icon (►), while others (in Blender) display both KS cue and submenu symbols for the hierarchical items. The end result in this case looks like this: Shift Ctrl O ►.

TOOLBAR

Toolbar is widget usually rectangular divided into rows and columns in order to form a grid cells. Each cell represents a command by either the use of an icon or with the combination of icon and text. For any extra information they use tooltips.

Tooltips are not visible all the time. To appear the users have to hover over the icon using the mouse pointer and after a few milliseconds a box appears (usually below the icon) that provides information about the command.

The information that tooltips provides varies from one application to another. I have identified the following types of information:

- The KS cue.
- The command label.
- A combination of both KS cue and command label.
- A description of the command's functionality.

One interesting observation is that there cases where a KS has been assigned to a command but the tooltip doesn't provide the KS cue . For example in the *Numbers* application the command Comment has as a KS the sequence ⇧⌘K. In the toolbar the tooltip of the command's icon provides only the description of the command's functionality without providing the KS cue .

It is important to note that I found cases where although a KS has been assigned to the command, the tooltip provided information about the command's functionality but not information about the KS cue . For example in the Numbers application Comment command has the ⇧⌘K as KS, yet the toolbar tooltip inform the users about the command's functionality without providing the KS cue.

Figure 3.5 shows two different toolbar examples along with the type of information the tooltip provides.

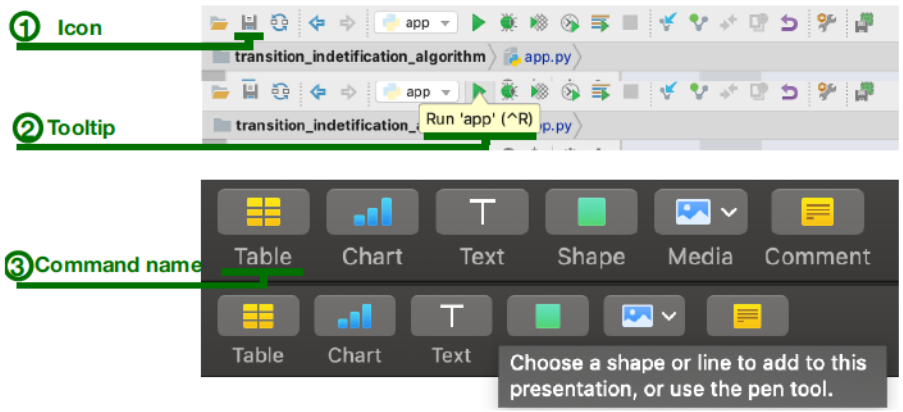


Figure 3.5: Two examples of toolbar layout. The top layout shows the toolbar from the Pycharm application. This toolbar shows only items and when the users hover over a button the name of the command along with the keyboard shortcut appears. The bottom layout shows the toolbar from Keynote application. This toolbar displays icons along with text for each command. When the users hover over an item a tooltip appears informing them about the functionality of the command

3.2.2 Relationship between the elements

Now, that I have described the different elements and their arrangements in linear menus and toolbars, I analyze the relations between these elements at a more abstract level providing opportunities for designs in the following chapters.

I aim to show that especially when it comes to the relationship between KS and the other two elements there is space for improvement.

3.2.2.1 Command name - Command icon

As I discuss in the previous sections both the command name and the command icon should convey successfully the systems' functionality. This fact can imply a strong semantic relationship between the two elements.

Quite often, in both linear menus and toolbars only one of the two elements[3] appear. This practice is probably due to the fact that both elements convey the same information i.e. the system's functionality.

In linear menus, the command label and the icon appear at the same time (when the menu opens). In toolbars, the icon is always visible. If the command label is not visible below the icon, users have to see the tooltip introducing a temporal cost.

When both available, the command label and the command icon are close to each other. In linear menus the command icon is on the left of the command label while in toolbars the command label is either

3 the command label or icon

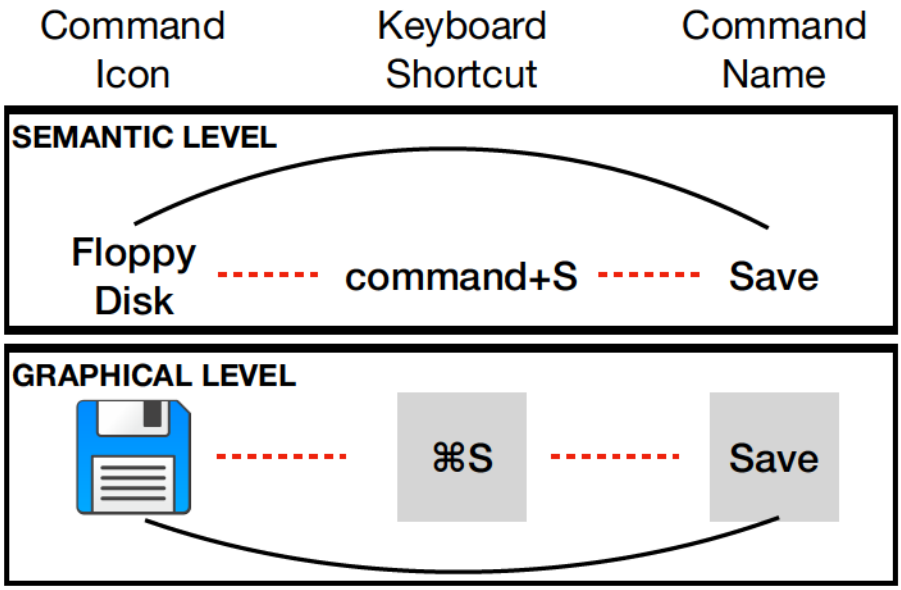


Figure 3.6: The analysis of the relations between the three elements highlighted that command name and command icon have a solid relationship (annotated with black solid line) while there are still opportunities to improve the relationship between the KS and the other two elements (annotated with the red dashed line).

below the command icon or in the tooltip (which is always close to the command icon).

3.2.2.2 *Command name - Keyboard shortcut*

Regarding the semantic relationship between the command name and the KS, design practices try whenever is possible for the command name to communicate the KS. Designers achieve this by using the first letter of the command as the hotkey of the KS key sequence. However, it is not always possible to follow the this guideline due to name conflicts.

In linear menus the command name label and the KS appear at the same time when the menu is activated. In toolbars if both the command label and the KS cue appear, then it depends where the command label is located. If the command label is below the icon then it is always visible, while for the KS cue to appear the tooltip needs to be activated.

In linear menus the distance between the command label and the KS cue is not fixed. It depends on the command label that has the largest length (in terms of number of characters). As a result commands with small names are far from the KS cue while commands with long names are close to the KS cue .

In toolbars if both command label and KS cue are available then they usually locate close to each other. If the tooltip contains both the command label and the KS cue then they are next to each other. If the command label is below the command icon then it depends on where the tooltip appears but it not far from the command label .

3.2.2.3 *Command icon- Keyboard shortcut*

Current design practices do not focus on the mental relationship between the command icon and the KS. It is interesting to note though that in some cases the icon successfully conveys the meaning of the command and the shortcut.

Let's consider the Bold command. In the english language layout the command icon (B) directly communicates the KS (Ctrl+B). In the french layout though the Bold command is renamed to Gras and the KS is Ctrl+G but the command icon is still B. So this incidental mental relationship depends on the language that layout uses.

Another interesting example is the Cut command. A typical icon for the Cut command is the ✂ and the corresponding KS is the Ctrl+X. Although it is not as direct as the Bold command, the shape of the icon strongly resembles the letter of the KS. This relationship though can break easily if designers choose another icon like this one ✂ [4].

In linear menus the command icon and the KS cue appear at the same time when the menu is activated, given that the command icon appears in the menu. However, in toolbars the users have to wait until the tooltip appears to see the KS cue , given that the tooltip includes the KS cue .

In linear menus the distance between the command icon and the KS cue depends on the length of the command name label. On toolbars the KS cue appears in the tooltip when it is available and with the extra temporal cost.

3.2.3 Discussion

The discussion in this section reveals that there is room for improvement for the semantic and graphical relationship between the command name , the command icon and the KS (fig. 3.6). In this thesis, I focus on the relations between the keyboard shortcuts and the other elements at the semantic and graphical level.

There is some effort regarding the semantic relationship between the KS and the command name. But these practices do not work well when there is a conflict between the name of commands.

The semantic relationship between the command icon and the KS is less explored. With the exceptions that I mentioned before, which are not necessarily intentional, there aren't currently guidelines for creating such a relationship between the two elements.

Similarly the graphic appearance of the KS cues are not consistent in command selection methods. Improving these relationships can potentially lead to better KS adoption.

To summarize, in this section I discussed the design practices of traditional layouts focusing on the semantic and graphic relationship

4 The application SketchUp uses a similar icon

between the three elements. Improving these relationships can potentially promote the KS adoption. Current HCI researchers however have focused more on proposing interaction techniques that provide feedback/feedforward related to KS. In the next section I am going to focus on these techniques.

3.3 INTERACTION TECHNIQUES TO PROMOTE SHORTCUT USAGE

Several interaction techniques have been proposed to improve performance while using a Recognition-Based or a Recall-Based method and to help users transition from one method to another. Since the main focus of this thesis is to help users transition to shortcuts we are going to focus on the interaction techniques that aim to help users to transition to shortcuts.

We have grouped these techniques into three categories : 1. techniques that aim to help users to learn the shortcut, 2. techniques that aim to raise the awareness of the shortcut and 3. techniques that aim to inform users about their future performance with the keyboard shortcut.

3.3.1 Learning the Shortcut

As we saw in the previous chapter in order to help users transition to a new method, the system should make the method easy to perform. In our case, this means that the system should help users learn the keyboard shortcuts. Here, we discussed the various techniques aiming to achieve this goal.

A part of the current literature has focused on creating easy to remember shortcuts. In particular for the case of Keyboard shortcut (KS), Green et al. [74, 75] examined the impact of consistency, mnemonics, and congruence when creating artificial languages. Within this scope, they tested these principles in keyboard shortcuts. They suggest that using modifiers consistently for the same semantic actions (e.g., using Ctrl for doing an action and Shift for reversing the action) and using sensible letters for the commands (e.g., using T for a command that has to do with text) can help users learn the KS. In the same spirit, Walker et al. [174] proposed a set of KS rules. They assigned each modifier to a family of functions (e.g., ALT for all the delete functions). Depending on the commands they used one or two letters, for example, they used ALT BC for "delete back a character." For the keybinding, they followed the order of Verb-Adjective/Adjective-Object. [22] proposes a way to design KS mappings without using modifiers. Instead, they

use two letters. The first letter corresponds to the menu category that the item belongs to and the second letter corresponds to the item itself.

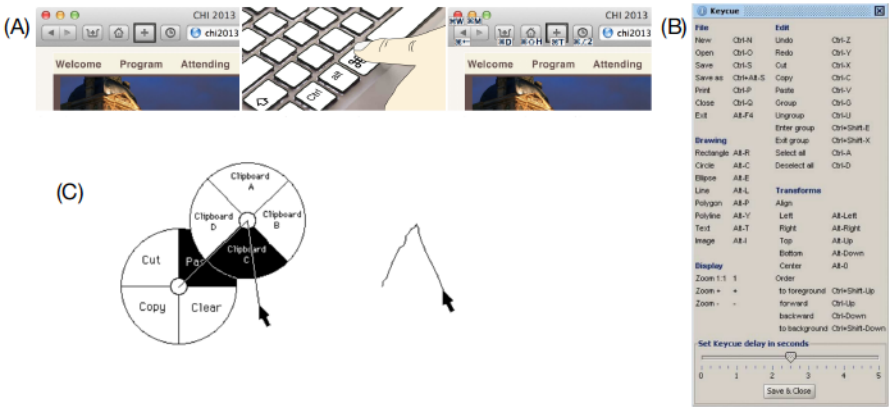


Figure 3.7: (A) ExposeHK [107] overlays the KS on top of the Recognition-Based methods when a modifier is pressed, (B) Keycue [157] displays a pop-up window with the relevant KS when a modifier is pressed, (C) Marking menus [95] display a pie menu in novice mode to help users learn the gesture.

Another part of the current research focuses on ways to help users learn the KS regardless of the mapping. ExposeHK [107] (fig. 3.7-A) is an interaction technique to help users learn the keyboard shortcuts through physical rehearsal and to browse the other KS. ExposeHK achieves this by overlaying the relevant KS on the Recognition-Based mehtods. For example, let's assume that a user presses a modifier. For all the commands in the toolbar that have this modifier the KS is going to be overlayed on the toolbar item. As a result, the users would know which are the keys that they need to press to invoke the keyboard shortcut. Keycue [157] (fig. 3.7-B) follows the same principle but they use a pop-up window that displays the relevant KS. For Stroke Shortcut (SS), marking menus [95] (fig. 3.7-C) allow users to choose commands either by using a pie menu (novice mode) or by making a stroke gesture (expert mode) to the direction of the position of the command. The main characteristic of the marking menus is the users have to perform the same physical movement in both novice and expert mode.

A family of interaction techniques focus on input devices to enrich the ways we can select a shortcut. For example, Xerox Star keyboard [24] and Microsoft Office Keyboard [27] (fig. 3.8-A) have dedicated buttons that are mapped to specific functionality. Although this solution is simple and offers a direct 1:1 mapping, it is not scalable for a large number of commands. GestAKey [146] (fig. 3.8-B) augments current keys of the keyboard to recognize touch gestures. GestAKey keys can identify the location and the motion of the finger. Therefore, users can select commands with simple gestures, e.g., swipe up on the letter "S" to get a screenshot.

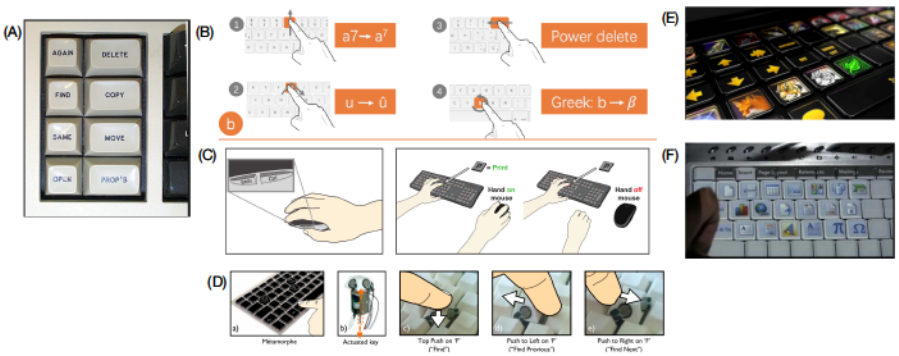


Figure 3.8: (A) Xerox Star keyboard [24] adds extra buttons for common commands like Copy, (B) GestAKey [146] keyboard can recognize gestures , (C) CtrlMouse [127] adds two extra buttons in the mouse while ToucCtrl [127] recognizes the Ctrl modifier when the users place their hand on the mouse, (D) Métamorphe [17] augments the key functionality so it can support more ways to input a command, (E) Optimus keyboard [11] and (F) Touch display keyboards [27] have keys that contain small screens which can be used to promote keyboard shortcuts

Métamorphe [17] (fig. 3.8-D) augments current keyboards with haptic and visual feedback. In particular, when users press a modifier the keys that correspond to this modifier are raised. Additionally, the keys can be pushed from all the sides. These attributes can afford different gestures that can be mapped to various commands thus creating a more possibilities for the designers. CtrlMouse and TouchCtrl [127] (fig. 3.8-C), users can select the Ctrl and Shift modifiers using the mouse. In particular, CtrMouse uses the two extra buttons that some mouse have on the side to map the Ctrl and the Shift and TouchCtrl triggers the Ctrl when users place their hands on the mouse. With these two techniques, users can invoke keyboard shortcuts more easily. Other keyboards focus on enhancing the output information they provide. Touch display keyboards [27](fig. 3.8 (F)) and the Optimus [11] (fig. 3.8 (E)) have keys that contain small screens. These screens can show command icons or relevant information so that the users can select commands more efficiently.

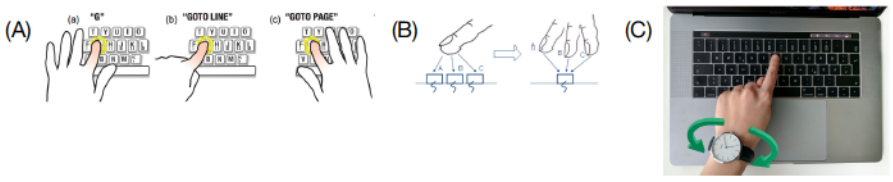


Figure 3.9: (A) FingerSense [175] recognizes the finger that press the button and executes the appropriate command, (B) Finger aware shortcuts [188] detect the hand , the finger the posture , (C) [30] can recognize wrist rotation

Another family of techniques combines gestures movements with shortcuts. FingerSense [175] (fig. 3.9-A) aims to express to improve buttons expressiveness by mapping different commands to the same button depending on which finger clicks it. So for example pressing the button with the index finger will invoke event A while pressing it with the thumb invoke event B. Finger aware shortcuts [188] (fig. 3.9-B) technique detects the hand and the finger that is used to press a key as well as open and close hand postures. By identifying those attributes, they can assign multiple commands in a single key press. To make the system work they used the built-in camera along with a reflector as well as an algorithm to recognize the hand, the finger, and the hand's posture. Buschek et al. [30] (fig. 3.9-C) have a similar approach, but they use the wrist rotation to generate new shortcuts instead of the finger, hand and hand's posture. They use a smartwatch to detect wrist movements.

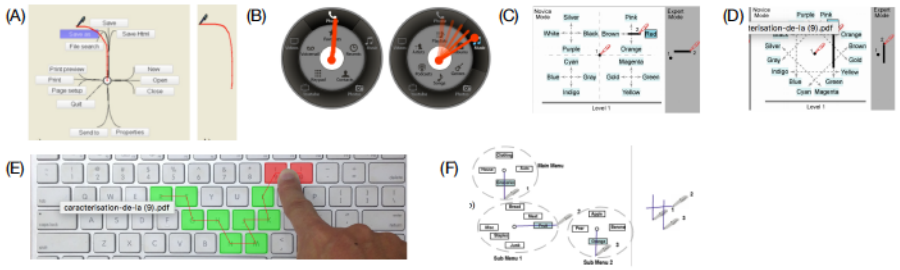


Figure 3.10: (A) Flower Menus [12] can support up to 56 commands, (B) Wavelet menus [55] create a visual perception of menu hierarchy , (C) Polygon [186] and (D)Zone menus [186] increase the breadth of the menu by considering the orientation and the starting point of the stroke, (E) GestKeyboard [185] supports strokes on the keyboard, (F) Zhao et al. [187] proposed a variation of marking menus to support hierarchical lists

Over the years many variations of the marking menus have been proposed to increase the number of commands that marking menus can support. For example, [187] (fig. 3.10-F) proposed a variation to support hierarchical menus using a series of inflection free simple marks. In a similar spirit, Flower Menus [12] (fig. 3.10-A) can contain up to 56 items by supporting not only straight gestures but also curved ones. Wavelet menus [55] (fig. 3.10-B) use a visual design based on a stacking metaphor to create the visual perception of the menu hierarchy. Wavelet menus also support a pre-visualization of the submenus thus better supporting navigation in a hierarchical structure. Finally, Polygon and Zone menus [186] (fig. 3.10-C,D) increased the breadth of the menu by considering the orientation of the stroke as well as the starting point of the gesture.

GestKeyboard [185] (fig. 3.10 (E)) takes another approach and uses the keyboard to recognize stroke shortcuts. The users' stroke multiple keys in a similar way they are stroked in touchscreens so that they can perform numerous actions. One of the main advantages of GestKey-

board is that it doesn't require any additional hardware because it uses a gesture occurrence classifier algorithm. Additionally, GestKeyboard supports modeless switching between performing gestures and typing to the keyboard.

3.3.2 Raising Awareness

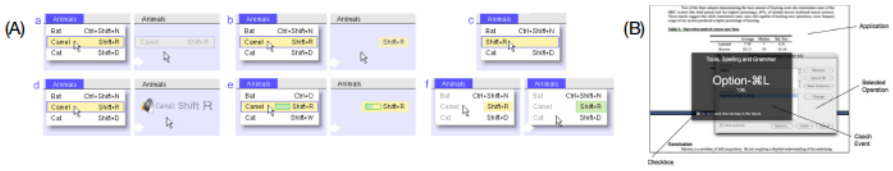


Figure 3.11: (A) Grossman et al. [77] proposed several menu designs to promote keyboard shortcuts, (B) Hotkey Coach [94] shows a transparent window that contains the KS when the users select a command using the menu or the toolbar

Some techniques use feedback to make users aware of the KS availability. For example, Grossman et al. [77] (fig. 3.11-A) proposed several menu designs to inform the users about the associated keyboard shortcut. The types of feedbacks varied from showing the corresponding KS after the users used the menu to auditory feedback and to the most extreme case of not allowing the users to use the menu at all thus forcing them to use keyboard shortcuts. Hotkey Coach [94] (fig. 3.11-B) displays a semi-transparent window whenever the users select a command with the menu. This window informs the users about the keyboard shortcut they can use to perform the same action and it gives them the opportunity to practice with it.

3.3.3 Perception of Future Performance

Figure 3.12: Skillometers [108] informs the users about their performance with the current method they use and the potential performance with other methods

Skillometers [108] (fig. 3.12-C) takes another approach to help users to switch to keyboard shortcuts. Two key factors to affect the users' decision to switch to more efficient strategies are motivation and their

perception of the future performance of the new technique [39, 143]. Building on these factors Skillometers offers a series of widgets that informs the users about their current performance when selecting a command and what is their potential performance if they choose the same command with a keyboard shortcut.

3.3.4 Discussion

To summarize current interaction techniques that promote shortcut usage aim to help users learn the shortcuts through various ways. Some aim to make the shortcut mapping easier to remember (e.g. [75, 174], others by finding intuitive ways to input the shortcuts (e.g. [16, 30, 185]. It should be noted that techniques that aim to raise awareness [77, 94] also help the users learn the shortcut. For example the first time the window of Hotkey Coach appears it informs the users about the corresponding KS, the rest of the times it will appear it helps the users learn the KS.

It is interesting to note that techniques that focus on the interface design (e.g. [77, 94, 107, 108, 143, 158]) they either focus on providing feedback or feedforward depending on the users actions. They do not discuss though the spatial and the semantic relationship between the command, the icon and the shortcut.

3.4 CONCLUSION

This chapter discussed the various methods that current applications support to select commands. Keyboard and stroke shortcuts are more efficient than other methods like menus, toolbars and ribbons but many users do not use them. Current interaction techniques that promote shortcut usage utilize feedback or feedforward. The information that they provide aims to raise shortcut awareness, help with the learning process and inform the users of the shortcuts' benefits. These interaction techniques however do not focus on the relationships between the command name, the command icon and the Keyboard shortcut (KS). One of research goals is to explore and improve these relationships. Especially in the case of KS, I will explore how KS adoption can be improved by improving the relationship between the KS and command icon (chapter 4) and between the KS and command name (chapter 5).

Part II

REVISITING CURRENT LAYOUT DESIGN PRACTICES

4 EXPLORING THE COMMAND ICON - KEYBOARD SHORTCUT RELATIONSHIP

Emmanouil Giannisakis, Gilles Bailly[1], Sylvain Malacria[2], and Fanny Chevalier[3]. 2017. IconHK: Using Toolbar button Icons to Communicate Keyboard Shortcuts. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17). ACM, New York, NY, USA, 4715-4726. DOI: https://doi.org/10.1145/3025453.3025595

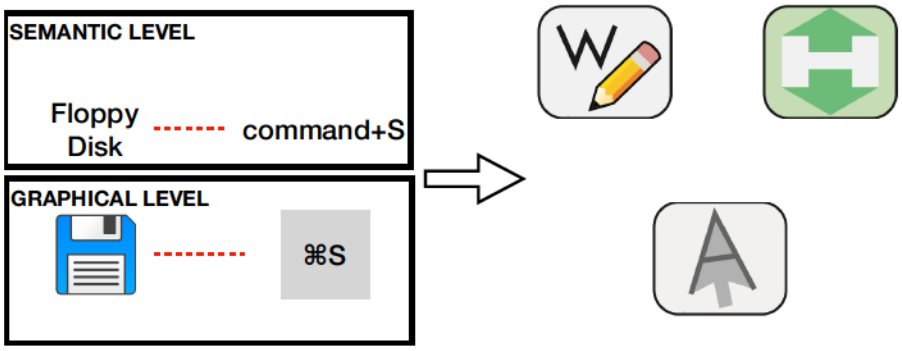


Figure 4.1: This chapter proposed a novel design perspective to design command icons that visual blend the Keyboard shortcut (KS) cue . The result icons can then communicate the command name along with the KS (*semantic relationship improvement*. Furthermore, the resulted icon brings together the KS cue with the command icon (*spatial relationship improvment*) and maximizes it's exposure.

This chapter focuses on the relationship between the icon and the KS on a mental, temporal and spatial level. I will propose a novel perspective for designing command icons called IconHK (fig. 4.8) which improves the way that command icons convey the KS.

With IconHK, we (I and my co-authors) aim at maximizing the exposure of the KS by visual blending it with the command icon. The resulted icon can inform the users about the KS without the use of tooltips thus making it always visible (improvement on temporal level) and minimizing the distance between the KS and the command icon (improvement on representation level). Finally as we are going to show with just the right icon (especially the objects that compose the icon) users can potentially recognize the KS even when it is not explicitly shown (improvement on mental level).

1 gilles.bailly@upmc.fr
2 sylvain.malacria@inria.fr
3 fanny@cs.toronto.edu

The chapter has the following structure First it discusses the IconHK approach by elaborating on the basic ideas behind this novel perspective.

Afterwards it discusses two user studies we conducted to evaluate IconHK. The first study examines the benefits of IconHK for end-users by assessing the hotkey retrieval effectiveness of the different strategies while the second study provides insights from professional designers on the practicality of the IconHK approach. Finally it presents a tool to help designers find inspiration to augment icons with the IconHK principle called IconHKMaker.

4.1 THE ICONHK APPROACH

This section explains the IconHK approach. The first part presents the overall design basics for creating enriched icons. The second part introduces the notion of *magnification* as the factor that determines the blending ratio between the traditional pictograph of a button and the most explicit and legible representation of the keyboard shortcut symbol.

In the following, we refer to *button* as an interactive control to execute a command, whose *icon* occupies a bounded physical space of a button (typically a toolbar button); *hotkey* as the character of a keyboard shortcut (which is generally a letter), and *modifiers* as its modifier key combination (respectively S and ctrl for the ctrl+S keyboard shortcut of Save) of a command; *pictograph* as the pictorial element of a button's icon that conveys the meaning of the command (e.g. the floppy disk for a Save command button); and *symbol* as the embedded hotkey letter (e.g. S for the ctrl+S keyboard shortcut of the Save command).

4.1.1 Embedding keyboard shortcuts into toolbar buttons

Our initial goal consists in embedding visual cues conveying the hotkey and modifiers in a button, which means embedding the symbol and indicators about the modifiers while preserving the aesthetic and legibility of the pictograph.

4.1.1.1 Conveying the hotkey

Inspired by logo design practices [148], we propose three strategies to embed a given symbol in a button while preserving the pictograph clarity: empty space, positive space and negative space as follows.

Empty space (fig. 4.2) consists in leveraging the blank space in or around the pictograph to display the symbol. For instance, only 23% of the pixels of the icon are devoted to the pictograph, which leaves enough room at the top-left of the button, should the hotkey symbol

be integrated (P). The icon has a large empty space within its pictograph which is sufficient to insert the symbol C. The strategy is limited to icons with few painted pixels, or containing large uniform areas. Several icons such as or do not afford large enough of a space to incorporate a symbol as is.

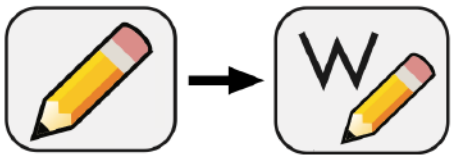


Figure 4.2: Empty space leverages the blank space in or around the pictograph to display the letter

Positive space (fig. 4.3) consists in revealing the symbol from the silhouette or the most salient features (i.e. edges) of the pictograph. This strategy is often employed by logo designers to seamlessly blend text and images in a single visual or use objects of particular shapes to evoke letters [148]. Transposed to icons, this strategy is best illustrated with the scissor icon of the Cut command, whose pictograph's shape resembles the X letter of the corresponding shortcut. It is worth noting that, in many applications, communicating the hotkey does not seem to be a primary objective. Probably for aesthetic reasons, different orientations of scissors are common, e.g. , which makes the perception of the hotkey more difficult. Ideally, the symbol derived from the pictograph's overall shape or edges should be as straight as possible to ensure that the augmented icon best communicates both the meaning and the hotkey. The icon is another example where the hotkey Q or O can easily be derived from, but, similar to empty space, not all icons qualify for this approach. Some pictographs do not easily support an encoded symbol through edge accentuation, such as .

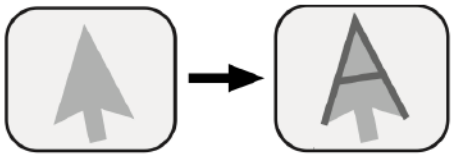


Figure 4.3: Positive space leverages the silhouette or the most salient features (i.e. edges) of the pictograph to display the letter

Negative space (fig. 4.6) consists in exploiting the open space around an object to disguise a letter. This visual effect— popular in logo design [148]—builds on figure-ground ambiguity that creates a visual that affords two alternative viewpoints, a common illusion technique that stems from Gestalt's principles [167]. A well-renowned example is the Rubin's vase [138] (section 4.1.1.1) where the white positive space forms a vase, while the black negative space forms two faces about to kiss. Transposed to icons, closed letters such as D or O can be

Figure 4.5
Rubin's vase illusion [138].

easily dissimulated in the negative space of an icon whose pictograph resembles a round shape. In contrast, the illusion becomes more difficult to achieve with open letters (e.g. 'J', 'E'), since pictographs are usually not confounded with icons borders (hence the negative space cannot result in an open letter). Another cognitive mechanism can help in these cases: the Gestalt principle of closure, that refers to our mind's tendency to perceive complete forms even if a picture is incomplete. This mechanism is well illustrated by the H letter that the can evoke. While it does not exactly correspond to the pictograph's negative space, 'H' can be deduced from the global pictograph's shape through closure.

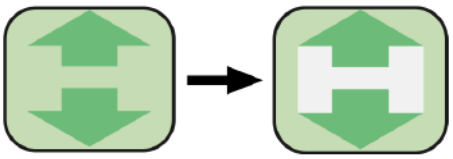


Figure 4.6: Negative space the open space around an object to disguise a letter

These three approaches offer a wide range of possibilities to embed the hotkey symbol into existing icons. They can also be leveraged to guide the design of novel icons in an application, and decide on best shortcuts hotkeys.

4.1.1.2 *Embedding Modifiers*

Embedding the sole hotkey symbol in an icon is sufficient when the keyboard shortcuts do not involve modifier keys (e.g, switching tools in Adobe Photoshop, or Final Cut) or when the shortcuts consistently involve the same modifier, typically ctrl or ⌘. When different modifiers are involved in the same application, further indications are necessary.

Embedding a graphical or textual representation of modifier keys can dramatically increase the complexity of the icon and affects its readability. To generate possible visual encodings of modifiers, we elicited ideas from four colleagues, all HCI experts not involved in the design of IconHK. We asked them, individually, to sketch on paper *"visual encodings that can be displayed on a toolbar icon to convey the modifier keys used in its keyboard shortcut "*.

One frequently proposed idea consists of mapping each modifier key to a square located at one of the four corners of the button, where a square is filled when the corresponding modifier is used by the shortcut and empty otherwise (shortcuts never involve more than four modifier keys), as illustrated in fig. 4.7. This design was proposed for its visual consistency, simplicity and because it capitalizes on spatial memory (i.e. the same modifier is always associated with the same corner). One of the solicited experts motivated: *"each modifier key can be stably*

mapped to a corner of the button, depending on its physical position on the keyboard: alt at the bottom-right, ctrl bottom-left and shift top-left." The notion of reusing keyboard layout to foster spatial memory was recurrent in other designs not involving corners.

Depending on the look-and-feel of the buttons to augment with IconHK, different visual indicators can be envisioned. Our colleagues mostly reasoned with squares; in Figure fig. 4.7, we used quarter-circle instead of squares in order to minimize the visual space occupied. This design also affords a richer status encoding, that can further help recognition and recall: when a user presses a specific modifier key, the corresponding corner is highlighted on all toolbar buttons in order to help users discover this mapping (e.g., the bottom-left corner is highlighted in blue upon Ctrl press in Figure fig. 4.7, right).

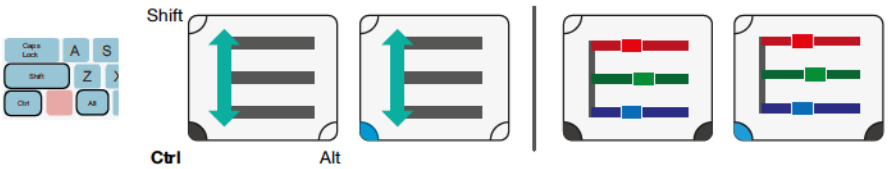


Figure 4.7: Example of icons augmented with iconHK that embed modifiers using the keyboard-location inspired mapping: alt at the bottom-right, Ctrl at the bottom-left, shift at the top-left. The commands *equal line space* (left) and *RGB colors* (right) respectively have Ctrl + E and Ctrl + alt + E as keyboard shortcuts. Corners are filled in dark to convey modifiers involved in the shortcuts (bottom-left corner only for *equal line space*, two bottom corners for *RGB colors*). The bottom-left corner is highlighted in blue when Ctrl is pressed to provide feedback.

4.1.2 The magnification continuum of IconHK

While we assume that modifiers can be displayed at all times without impacting buttons' readability, a compromise might have to be found for the symbol. It is not always fortunate, nor even possible, to legibly display the symbol, especially when doing so visually alters an icon to a point the pictograph is difficult to recognize, or simply for aesthetic reasons.

To overcome this issue, we introduce the notion of magnification continuum, as the entire spectrum of representations resulting from a progressive morphing from the pictograph representation, to the symbol only (fig. 4.8). At one end of the continuum, the button conveys the meaning, as found in most applications. At the other end, the button communicates the hotkey, as in Hopper Disassembler [81]. This latter strategy is seldom used and only evocative to knowledgeable users, as users who are not aware of the shortcut mechanism or key combination might be confused because they do not understand the meaning

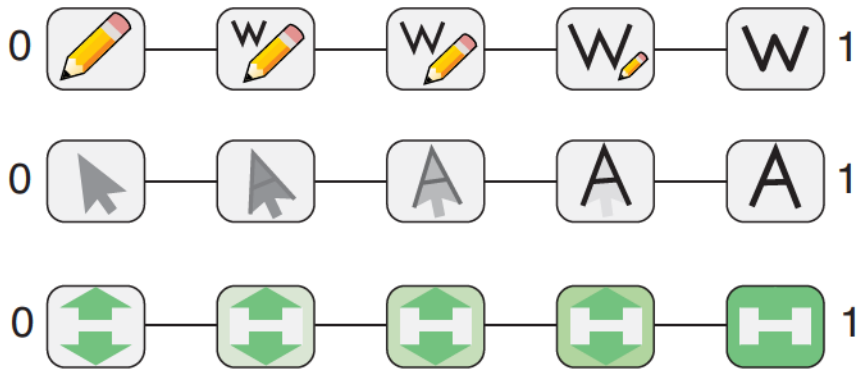


Figure 4.8: Variations of toolbar buttons on the IconHK continuum for the *Pencil*, *Move* and *Expand vertically* commands. From 0 to 1 on the continuum: *Pencil* scales down while a W scales in, exploiting the empty space; *Move* rotates and fades out the pointer while revealing a A overlaying the edges; *Expand vertically* changes the background color of the button to emphasize the H symbol in the negative space.

of the icon. IconHK provides static and dynamic alternatives that lie between these two extremes.

4.1.2.1 Formal definition

The magnification continuum ranges from 0 to 1, where 0 means that the button displays the pictograph only, and 1 means that the button displays the symbol only. All representations in between are a blend of the two (Figure 4.8).

Toolbar buttons in an application can be set a default static representation from anywhere in the spectrum at a value that the designer judges offers the best tradeoff, e.g., the mid-point of the spectrum. Or, the whole or part of the contiuum can be leveraged in a dynamic manner, e.g., buttons can smoothly animate between a representation closer to the pictograph representation to one closer to the literal symbol representation of the shortcut hotkey when hovering over (we discuss different interaction design options in a later section).

To formalize, let us consider a toolbar button enriched with IconHK. We define two stable states noted $M_{\sqsubset}$ and $M_{\sqsupset}$, that correspond to the two bound states along the continuum between which the button can vary (e.g. pictograph as the default state, and slightly revealed symbol as the hovered over state), and let $M_{|}$ be the current state of the button. We have:

$$0 \leq M_{\sqsubset} \qquad M_{\sqsupset} \leq 1 \qquad M_{\sqsubset} \leq M_{|} \leq M_{\sqsupset}$$

The values for $M_{\sqsubset}$ and $M_{\sqsupset}$ are to be defined by the designer, and may be different across buttons depending on the embedding strategy and interaction design. For example, a designer may choose to set all

icons enriched with empty space strategy with a $M_{\sqsubset} = M_{\sqsupset}$ value just high enough for the symbol to be visible (e.g. for the pencil command of Figure 4.8, $M_{\sqsubset} = 0.2$ corresponds to). For all other icons, she may set $M_{\sqsubset}$ to zero and $M_{\sqsupset}$ to one (e.g., for the expand vertically icon, $M_{\sqsubset} = 0$ corresponds to and $M_{\sqsupset} = 1$ corresponds to), betting on the users' capability to associate the pictograph with the corresponding letter while providing the opportunity to hover over the button for recall. Similarly, $M_{|}$ can vary independently for each button, depending on users' interaction. We discuss how these parameters can be leveraged to support different design scenario in a further section.

4.1.2.2 *Magnifying along the continuum: symbol saliency*

As we move along the magnification continuum, the symbol is progressively revealed until it becomes fully legible by augmenting its saliency, an effect that can be achieved with different transformations such as rotation, translation, scaling as well as manipulation of the opacity or color of the foreground and background. The approaches depend on the magnification strategy and designers' preferences.

For instance, the empty space strategy leverages pixel areas in two ways. When the symbol is to be embedded in a pixel zone unoccupied by the pictograph, augmenting its saliency can be achieved by increasing the symbol's size while reducing that of the pictograph (Figure 4.8, top line), resulting in sort of a swipe transition. When the empty space corresponds to an homogeneous area within the pictograph (e.g.), instead of reducing the pictograph's size, we make it grow while also increasing the symbol's size, as if zooming on the area containing the symbol until it occupies the whole area.

Both the positive and negative space strategies leverage the pictograph features to reveal a symbol. An approach consists of adapting opacity and color of the symbol and pictograph so as to progressively fade in the symbol while fading out the pictograph. For the positive space strategy, this amounts to reducing the opacity of the pictograph except from the salient features that delineate the symbol, which, conversely, are emphasized. For negative space, the idea consists in progressively reinforcing the contrast between the positive and negative space, and further extend the positive space to create a closure (see Figure 4.8, bottom line). Additional affine transformation might also be used to realign the symbol. More advanced transformations might use semantic elements of the pictograph to reveal the symbol (e.g. slider thumbs translate to form the 'E' symbol in Figure 4.9).

4.1.3 IconHK dynamic behavior

IconHK affords many possibilities in terms of icon behavior. We imagine three cases when transitions from pictograph to symbol (and vice versa) could be relevant.

1. *Regular reminders*. To give users awareness of the special nature of the toolbar icons, and prompt hotkeys as a reminder, an application could emphasize symbols of the whole toolbar at launch for a few seconds, before switching back to the more traditional pictographs. This operation could also be triggered every now and then to foster awareness and recall.

2. *Adaptive saliency*. The magnification level of icons could evolve depending on application usage. For instance, the $M_{\sqsubset}$ and $M_{\lrcorner}$ values can be set to vary depending of user's expertise regarding a certain command (Figure 4.9). Typically, the more the user selects a command using the mouse, the closer $M_{\sqsubset}$ gets to $M_{\lrcorner}$ for that command, thus encouraging hotkey usage. Conversely, the more a command is selected using its hotkey, the closer $M_{\sqsubset}$ gets to 0, since the user's behavior suggests she masters the hotkey and does not require a visual aid. Note that different strategies could be explored, for instance by increasing $M_{\sqsubset}$ value regardless of the modality used for selecting the command, thus not only increasing the saliency of the symbol, but also communicating how frequently a command is used, in a similar fashion as [45]. Another strategy could be to update $M_{\sqsubset}$ or $M_{\lrcorner}$ values as a function of the frequency of the command among a group of users.

3. *Interactive saliency*. The dynamic behavior of icons could also vary depending on user's interactions with the system at a much lower level, as for instance, when hovering over the icon or when pressing a modifier key, as described below.

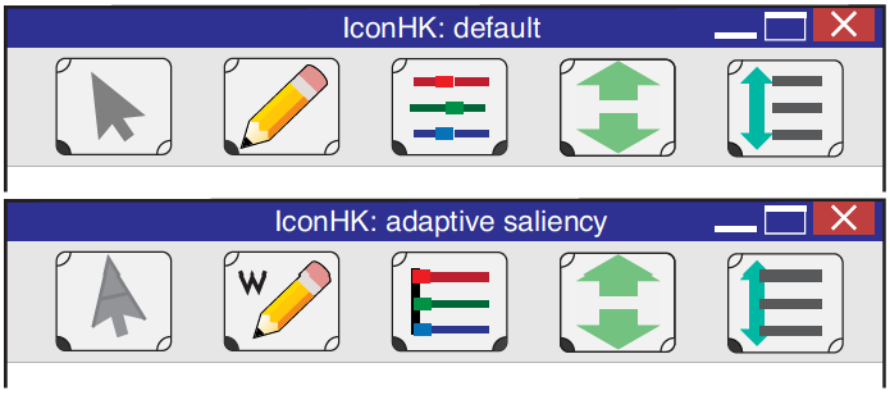


Figure 4.9: Example of a toolbar using IconHK. Top: by default ($M_{|}$), the magnification of all buttons is equal to $M_{\sqsubset}$: the hotkey symbol is not emphasized. Bottom: the individual $M_{|}$ of each button reflects user's command selection habits: the more frequently a command is selected by clicking, the higher the value of $M_{|}$ is.

Inspired by ExposeHK [107], $M_{\lfloor}$ values of all toolbar buttons can gradually increase to $M_{\urcorner}$ when the user presses a modifier key to better emphasize the symbols associated with this modifier (see Figure 4.10), and remain as such until modifiers are released, triggering a saliency decrease back to $M_{\sqsubset}$. This magnification provides a feedforward mechanism allowing users to identify the hotkey when initiating a keyboard shortcut. This feedforward mechanism is less intrusive than pop-up mechanisms (like ExposeHK) in that the symbol is seamlessly embedded within the icon, preventing occlusion, and gradual transition within the icon itself is less visually distracting than appearing pop-ups.

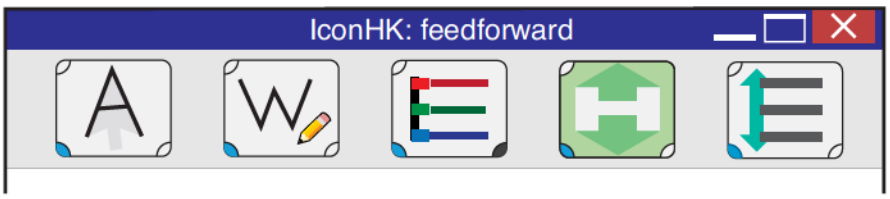


Figure 4.10: When the user presses a modifier key, the magnification factor of all buttons increases to $M_{\urcorner}$ to maximize the saliency of the hotkey when the user needs them the most.

Similar to [77, 94], IconHK can also be used to provide feedback, in complement to feedforward, when a command is activated using the mouse. For instance, every time a command is selected in such a way in the menubar, the $M_{\lfloor}$ value of its corresponding toolbar button gradually increases to $M_{\urcorner}$ and is maintained at this value for a given time laps (e.g. 500ms)—guaranteeing a maximal saliency of the keyboard shortcut during a short period—before it is progressively decreased back to $M_{\sqsubset}$. When a command is selected using the mouse in the toolbar, $M_{\lfloor}$ immediately increases to $M_{\urcorner}$ in order to magnify the symbol, and remains as is so long as the toolbar button is pressed, or until the command is activated (Figure 4.11). When activated, $M_{\lfloor}$ remains equal to $M_{\urcorner}$ for an additional given time before to decrease back to $M_{\sqsubset}$.

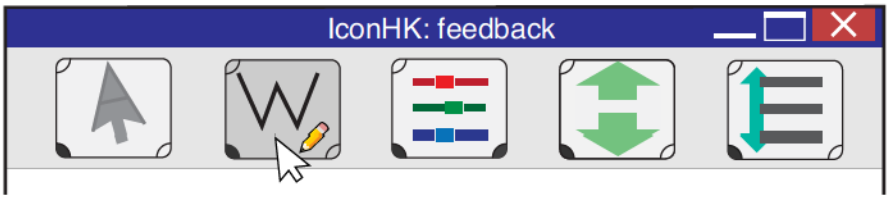


Figure 4.11: When the user presses on a toolbar button, its saliency instantly increases to $M_{\urcorner}$.

Overall, the transitions presented in this section are just one instance of possible dynamic IconHK behaviors, but other possibilities could be explored. Still, we recommend that transitions, especially when based

on user's interaction, are always animated to facilitate comprehension and foster learning, as well as maintain a high visual appeal.

4.1.4 Mnemonic mechanisms

Widespread icons are interesting to analyze from our perspective of using icons as mnemonic aids. We discussed that the scissor (Cut) could evoke the 'X' symbol. This pictograph was inspired by traditional practice in manuscript-editing whereby people would cut paragraphs from a page with scissors and paste them onto another document. It remains, however, unclear if the hotkey X was chosen because of its similarity with a scissor silhouette, or if the icon was designed to evoke the standard hotkey, or none of the above.

It is worth noting that X, C, V and Z hotkeys for the most common commands Cut-Copy-Paste and Undo were chosen to be all clustered together to facilitate sequence of operations. Which one of these hotkeys was first, if any, and whether such choice was guided by a mnemonic strategy (i.e. 'X' to recall the shape of the scissor? 'C' for Copy?) also remain obscure. The important lesson to retain is that there are several factors that come into play when choosing a hotkey, be them other mnemonic mechanisms such as using the first letter of a command to reinforce associations, or constraints that limit options for icon and hotkey design.

Formatting text icons are also interesting in that the formatting effect is directly illustrated within the icon, e.g. I for Italics or B for Bold. Here,'B' is both the first letter of the command name and the chosen hotkey in English language (ctrl+B). We did not find any reference allowing to claim whether 'B' is used as a reminder for the command name or for the hotkey. Yet, in French applications, the corresponding pair (G icon; ctrl+B shortcut), the icon refers to the command name ('G' for Gras) rather than the hotkey B.

4.2 REDESIGN OF THE PHOTOSHOP PALETTE

In this section, we revisit the Photoshop CC 2015 palette (Figure 4.12-top) to communicate available keyboard shortcuts with IconHK. Our goal was to respect the original icon style and thus make as minimal changes as possible to not confuse users already familiar with the traditional iconset. Our proposed design is illustrated in Figure 4.12-middle & bottom.

Several icons were very suitable to embed a hotkey symbol, e.g., squeeze the 'G' symbol within a bucket (Paint Bucket); rotate the eyedropper to insert a vertical 'I' (Eyedropper); emphasize the edges of the arrow to reveal a 'A' (Move). Interesting design cases are the Pencil and the Pen tools that both use the empty space to embed the

'B' and 'P' symbols respectively, while leveraging the semantic of the command (i.e. the tools draw the letters).

We found some rather obvious embedding cases could introduce confusion, e.g. the silhouette of the Dodge tool evokes a 'Q' whereas the hotkey is 'O'. We flipped the pictograph so that the handle is less suggestive of the 'Q' descender .

Figure 4.12-bottom shows symbols with a high level of magnification. In practice, the default magnification could be less or more salient (e.g. Figure 4.12-middle), and icons made responsive to users' interaction as previously discussed. Our intent with this example was to demonstrate that IconHK can successfully be incorporated to already existing toolbars, though some enriched icons may not be exemplary due to the strong constraints of keeping a design style close to the original. Obviously, designing iconsets from scratch, with the IconHK principles in mind, offers more flexibility, which can result in more cohesive and more aesthetically appealing iconsets.

4.3 STUDY 1: ICONHK AS A MNEMONIC AID

The primary goal of IconHK is to provide a visual aid for prompting keyboard shortcuts, so we conducted a user study to assess its potential as a mnemonic aid. More specifically, we investigated whether after a relatively short exposure to an IconHK icon with an intermediate level of magnification (i.e. $M_{|} = 0.5$) participants were able to recall the cor-

Figure 4.12: Possible adaption of the Adobe Photoshop CC 2015 interface using IconHK. Top: default toolbar. Middle & Bottom: the adapted toolbar integrating IconHK principles at two different levels of magnification. All buttons embed their associated hotkey, except the Blur button (water drop) which lacks a hotkey.

responding Keyboard shortcut (KS) only by looking to the pictograph ($M_I = 0$).

4.3.1 Participants

Twelve volunteers (1 female) participated in the experiment. Participants ranged in ages from 22-36.

4.3.2 Design

A within participants design was used. The independent variable was the strategy that was used to embed the letter inside the icon. In particular we used the three IconHK strategies (*Empty*, *Positive* and *Negative* space) and a *Control* condition (where the hotkey is not embedded in the icon in any way).

We used 4 commands per strategy for a total of 16 commands (see Figure 4.13).Every command had an icon and an associated keyboard shortcut. All keyboard shortcuts used the Ctrl modifier only, and a character key that was not the first or last letter of the command name [7, 77].

The experiment was divided in 4 repetitions of training and test phases (inspired by [21, 137]) . In each phase, the 16 commands were presented in a randomized order, with each command appearing exactly once.

In summary, each of the 12 participants performed 4 design strategies (Positive, Negative, Empty, Control) ×4 commands = 16 trials per test phase (4 laboratory tests +2 online post-tests) for a total of $12 \times 4 \times 4 \times (6 + 2) = 1152$ trials.

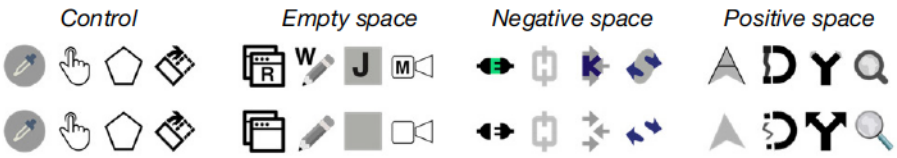


Figure 4.13: Icons used study 1. Top: icons used for the trainings (magnified for IconHK); Bottom: icons used for the tests (not magnified).

4.3.3 Procedure

Training: During this phase, the interface shows the 16 commands in the toolbar. Toolbar buttons relying on IconHK were magnified with a level of $M_I = 0.5$ so that the icon conveyed both the meaning and keyboard shortcut of the command. Participants were asked to select commands as fast and accurately as possible using their keyboard shortcuts. If they

did not know the shortcut, they could hover the corresponding button with the mouse pointer to reveal a tooltip displaying it.

Test: In this phase, the system displays unmagnified icons as visual stimulus, i.e. the letter was not printed for the empty space strategy and the positive/negative space was not highlighted any more. Participants were asked to execute the shortcut corresponding to the requested command as fast as possible. If they did not know the shortcut, they had to try to guess anyway. For Positive and Negative, this is a *retrieval* test as some visual cues remain in the icon. For Control and Empty, we did not expect participants to retrieve the shortcut from the icon, thus this can also be seen as a *recall* test.

Participants then performed online post-tests, replicating the test phase 24 and 72 hours after the experiment without being re-exposed to shortcuts (that is, participants did not practice the shortcuts after the 4th block of the initial test).

4.3.4 Results

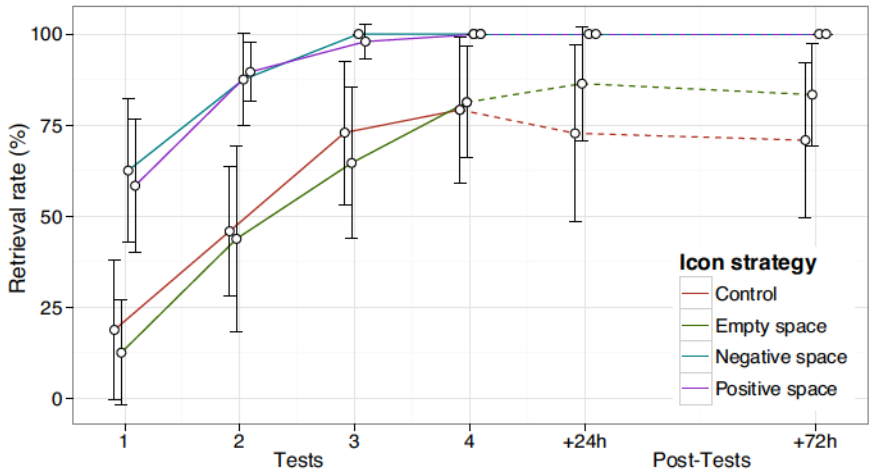


Figure 4.14: Percentage of recognition rates for each Icon family, for every block and retention tests (dashed)

We define the *retrieval rate* as the proportion of correct answers in the test phases. The average retrieval rate across the 4 initial tests was 54% CI[43,65] for Control, 50% CI[39,62] for Empty, 88% CI[80,94] for Positive and 87% CI[80,93] for Negative. It was 73% CI[59,87] for Control, 84% CI[76,95] for Empty, and 100% for both Positive and Negative for the 2 post-tests. Figure 4.14 shows that learning was consistent across blocks for all conditions, but quicker for Positive and Negative reaching almost 100% at the 3rd test and remaining stable until the 2nd post test, 72h after training. It also shows that Control was the only condition where the retrieval rate decreased between the last test block and the 72h retention test, changing from 79% back to 71%.

These results suggest that brief exposures were sufficient for participants to retrieve hotkeys with the Positive and Negative strategies, even 72 hours after the last training. These encouraging results motivated us to learn more about how IconHK principles might be included in designers' practice.

4.4 STUDY 2: INSIGHTS FROM DESIGNERS

We conducted a second study with professional designers to collect feedback on their impressions about, and possible difficulties with the IconHK concept, as well as gain insight in the design process for creating iconsets integrating IconHK principles.

4.4.1 Participants

Three professional designers (2 females) aged 28 to 29 were recruited. We refer to them as D1, D2 and D3 in the following. All have experience in both graphic design and logo/icon design (3, 5 and 6 years respectively).

4.4.2 Procedure

Each session started with the experimenter explaining the IconHK concept, i.e. the 3 embedding strategies and the magnification continuum. Participants were then instructed to design three iconsets, one per embedding strategy given the same set of 6 commands: *Save*, *Copy*, *Print*, *Edit path*, *Plot* and *Refresh*. We chose this set of commands to have a balance between most common commands (*Save, Copy, Print*) and more specialized ones (*Edit path, Plot, Refresh*), for which a particular icon metaphor may be less standard.

For a given IconHK strategy, the task was to create an iconset implementing the embedding strategy for each of the six commands. Designers were instructed to consider good icon/shortcut design practices: for each command, (1) choose a pictograph that best represent the meaning of the command and (2) a symbol (i.e. hotkey letter) that can act as a mnemonic for the command, as well as (3) maintain consistency across the entire iconset (i.e. graphic style). We left them free, however, to weight each objective as they saw fit to make the best design compromises for the whole iconset.

To facilitate, designers were free to sketch icons with their preferred tools. In the case of static sketches (pen+paper, or vector graphics), we asked them to provide at least 3 levels of magnification for each icon to illustrate the steps of the animation. We also invited them to draw

inspiration from a collection of icons (6 per command) that we curated to illustrate various pictograph metaphors as well as graphical styles.

Designers were encouraged to provide insights on their design process through a think-aloud protocol. The experimenter was present during the whole session and was taking notes. When deemed necessary the experimenter asked the designers for clarifications. The session ended with a semi-structured interview inquiring about their design decisions, impressions and the difficulties they encountered. Figure 4.15-left shows a selection of resulting designs (see chapter h for more designs).

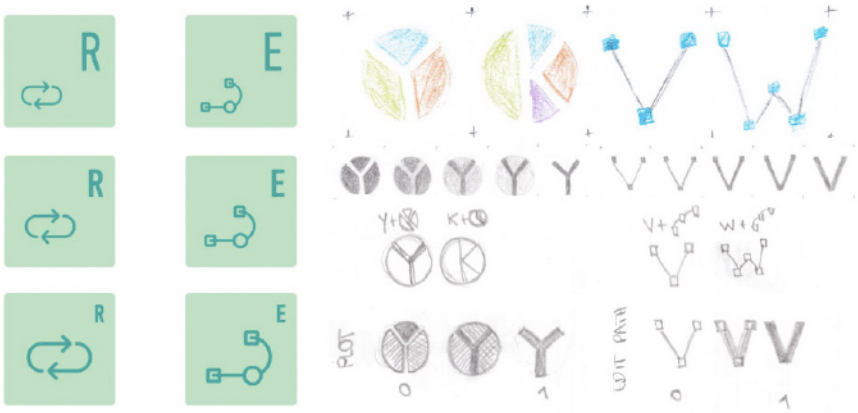


Figure 4.15: Left: Icons implementing the Empty space strategy from the first part of the study. Right: Positive and Negative sketches with different magnification from the follow-up study: *Plot*: A pie chart pictograph embedding either a 'Y' or a 'K'. *Edit Path*: A connected line embedding either a 'V'or a 'W'.

4.4.3 Results

All three designers felt more comfortable using a vector graphics software or paper+pencil over animation software or programming. Designers interrupted the session (afer 3h, 2h and 4h respectively for D1, D2, and D3) before completing all three iconsets, albeit no time constraint was imposed. We discuss here the key insights of our study:

Design process. It was interesting to observe how designers spontaneously made radical decisions regarding good icon/shortcut design practices. All of them primarily focused on consistency between icons, as they generally feel that a consistent iconset is more *"user friendly than an iconset that better communicate the information to the user,"* a mindset in contradiction to IconHK foundation. How they valued consistency had an impact on the choice of the strategies: they believed that mixing the embedded strategies in an iconset could be confusing to the user because the icons would not have the *"same identity."* After the experiment, informal discussions with HCI researchers working closely with designers confirmed that consistency is strongly anchored in de-

signers' practice. The choice for pictograph and hotkeys was lower-ranking, yet designers picked meaningful one. They favoured simple outline or glyph designs for the pictograph, drawing inspiration from the icons we provided and also from other examples from the web focusing mostly in minimalistic designs. Finally, they consistently chose the first letter of the command as hotkey, unless name collision occurred, where they favored the most frequent command and searched on the web which shortcut is generally used for the other command.

IconHK concept and feasability. Designers thought that the concept of IconHK is interesting and potentially useful for the users but expressed reserve about the feasibility of the technique. D3 found that *"this technique can benefit the users if done correctly."* D2 reported *"I can see how this can be useful for the end user and it poses an interesting design problem as well,"* further commenting that *"creating an iconset which is coherent, minimalistic and successfully communicate the meaning and the hotkey is a time and resource demanding task. I am not sure the results would be satisfactory."* which explains their difficulty completing the task.

Empty space. Empty space appeared to be the easiest strategy to implement. Indeed, spatially organizing the pictograph/hotkey requires less effort than creating an icon with positive or negative space. Moreover, it is the easiest strategy to maintain consistency across icons. For instance, each set of icons used the same position for the hotkey. Designers also preferred this strategy because it has a *"minimalist"* style which is a current trend in icon design.

Positive space. All designers tried to experiment with the positive space. D1 wanted *"to see if this strategy comes natural to [her]."* However, designers quickly came to same conclusion that it would require a considerable amount of time to produce satisfying results. The main reason is that consistency was their primary consideration. They explained that it would require to design their own font, i.e. designing letters that have the same look&feel and at the same time fits the pictographs of the iconset. In their opinion this task would require several workdays, but found the exercise interesting.

Negative space. This strategy was considered as the most difficult one because the pictograph and the letter might have completely different shapes. For example D1 explained that for the *Save* command, Floppy disk and 'S' are respectively the most frequent pictograph and hotkey. However, the 'S' letter is curvy while the floppy disk is mainly made of straight lines. *"Creating a variation of either the floppy disk or the letter is a complex procedure and the result might be unsatisfactory"* (D1). This difficulty is amplified if all icons must rely on the same strategy to maintain consistency.

4.4.4 Follow up: Focus on Positive and Negative space

Because the designers experienced too much difficulty to sketch Positive and Negative solutions in the given time frame, we conducted a follow-up study focusing only on these two strategies by simplifying the experimental design. Two designers, D3 and a novel designer (D4) having 6 years of experience in logo/icon design participated to this study. They were compensated 15$/h (total 2x4h=8h) for their design work. Participants were asked to create 6 icons (instead of 18): 2 positive space, 2 negative space and 2 empty space. We emphasized that consistency between icons was not mandatory, but asked for five levels of magnification for each command. All icons are attached in the supplementary materials. Figure 4.15-right shows some resulting sketches (see chapter h for more designs) .

Findings: Since D3 was already familiar with the IconHK concept she did not have any new comments. D4 confirmed that the Negative space strategy was the most difficult one, especially for complex pictographs. However she also mentioned that *"choosing a letter that takes advantage of the shape of the icon can help users to memorize the hotkey, especially for visual persons."* D4 used a responsive approach for the Empty space strategy. Responsive icons dynamically change their shape and level of details according to their size and are well suited approach to be combined with Empty space. Concerning Positive space, D4 first superimposed the letter shape with the pictograph to see how well she can use the pictograph edges, otherwise, she thought about possible morphing animations but this approach is more complex.

4.4.5 Discussion

We learn that (1) the designers appreciated the IconHK concept, especially Empty space for its simplicity and its compatibility with minimalistic styles; (2) Positive and Negative space are more difficult to apply but raise *"interesting design challenges"*; (3) consistency is a primary quality criterion. While there are multiple debates in HCI/design about consistency vs. efficiency (e.g. "The case against User Interface Consistency" [78]), which are not always opponents, this study confirms the current practice of ignoring keyboard shortcuts in icon design and the need to advocate for design solutions favoring the transition from novice to expert behavior. This study also suggests the need for a tool that assists the designers in creating icons with the IconHK concept.

For more icons that the designers produced please see chapter h

4.5 ICONHK MAKER

We implemented IconHK Maker, a tool to assist designers to create icons with embedded keyboard shortcuts. This proof-of-concept should neither be considered as a definite solution to the problem of IconHK design or as a means to substitute designers. Rather, we investigate how image processing algorithms can provide suggestions during the design process: the creative work still requires the designer's judgement and expertise. We present two algorithms to embed a symbol within an icon, one of which identifies the largest empty space to display the symbol, whereas the other analyses the pictograph's features to best fit the symbol in either the positive or negative space. Given an icon, a symbol and a collection of fonts, the algorithms provides a transformation (translation, rotation and scaling) indicating how to embed the symbol in the icon with one of the IconHK strategies.

Finding Empty space. Finding an empty space to display a symbol in a button can be formulated as the *maximum empty rectangle problem* [114]. Given a rectangle R and a set S of n points, the problem consists of finding a maximum area rectangle that is fully contained in R and does not contain any points of S. We implemented the algorithm described in [114], but further adapted it to ensure that the resulting rectangle is not too flat to maximize the space for the letter.

Finding Positive and Negative space. Here, the goal is to minimize the distance between the symbol and salient features of the pictograph (e.g. its edges that can be made in common with the symbol). The algorithm searches the maximal number of edges that can be shared between these two visual elements. This optimization process comprises four steps. First, we **extract the different paths** of the pictograph and discard small shapes then we **re-sample** the paths to ensure that the distance between two consecutive points is the same. Next, we **apply the Ransac algorithm** (Random Sample Consensus) [52] . At each iteration, this algorithm generates a random transformation (see below), estimates the distance between the transformed symbol and the pictograph and returns the transformation with the smallest distance (Hausdorff [83]).

We consider two rigid transformations: translation and rotation, as well as two non-rigid transformations: scaling and truncation, i.e. removing some paths in the letter. The idea behind this transformation is that a letter's shape is rarely fully defined within an icon. For instance, in the icons (A) and (E), the embedded letter is not complete. Our truncation transformation removes up to two edges.

Finally, we **apply the ICP algorithm** (Iterative Closest Points) [23], which aims at minimizing the distance between two sets of points. ICP is very efficient to find local optimization but does not perform well for global optimization because it is strongly sensitive to the initialization.

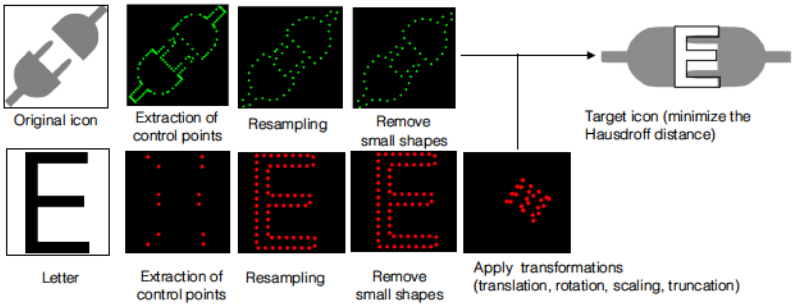


Figure 4.16: The algorithm to identify an hotkey symbol in the positive and negative space.

This is why we use both Ransac (global approximation) and ICP (local optimization) to match the two sets of points.

Interface and Interaction. Our current implementation (C++/Qt) integrates the described algorithms for SVG graphics. It provides three main functionalities: (1) users can edit the SVG icons by manipulating the control points. They can also/draw sketch over the icons. (2) Once users select a letter, the system suggests the best location to embed the letter for both of our algorithms. Suggestions are updated in real time while users are editing the icon. Finally, (3) the system can also suggest a letter given an icon. The algorithm iterates on each letter and keeps the ones with the best scores.

Figure 4.16 illustrates the intermediate steps and final result for embedding a 'E' symbol within a plug pictograph. Our informal tests with various icons and symbols suggest that computer graphics approaches could be a valuable support for recommendation. Yet, further refinements of the algorithm are still to be done to reach the necessary level of robustness of a usable tool. A thorough evaluation of the algorithms performance is beyond the scope of this work.

4.6 CONCLUSION

In this chapter, I discussed a novel approach to design toolbar button icons that aim to communicate the corresponding keyboard shortcut. The main goal of this work is to reinforce the relationship between the command icon and the Keyboard shortcut (KS) both in a semantically and spatially. Although from the study of the designer indicated that this approach adds another level of complexity, they can benefit the users and help them to adopt KS.

In the following chapter we will continue exploring how to improve the relationship between the KS, command icon and the command name . We will focus specifically on the improving the relationship between the command name and the KS.

5 EXPLORING THE COMMAND NAME - KEYBOARD SHORTCUT RELATIONSHIP

Emmanouil Giannisakis, Evantia Dimara [1], Gilles Bailly[2], Annabelle Goujon[3]. Revisiting Linear Menu Layout to Promote Keyboard Shortcut Usage. (to be submitted).

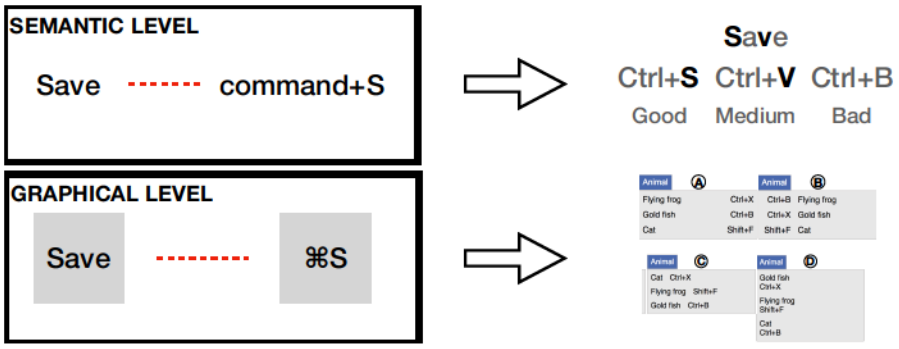


Figure 5.1: In this chapter I first explore how the command name can better communicate the Keyboard shortcut (KS) (*semantic relationship improvement* by considering the letter of the command name and the KS hotkey. Afterwards I explore different placements of the KS cue in the linear menu that aims to bring closer the command label and the KS cue *spatial relationship improvement*.

This chapter aims to improve the relationship between the command name and the KS. To achieve that we (me and my co-authors) are going to discuss possible command name - KS mapping strategies (improvement on mental level). Then we are going to discuss possible placements of the KS cue in linear menu that aim to minimize the distance between the KS cue and the command label (improvement on representation level).

Finally we conduct three user experiments. The first investigates 3 different command name - KS mapping strategies can affect KS adoption. The other two investigate novel menu layouts that change the position of the KS cues in linear menus.

1 evanthia.dimara@gmail.com
2 gilles.bailly@upmc.fr
3 annabelle.goujon@univ-fcomte.fr

5.1 REVISITING THE COMMAND NAME – KEYBOARD SHORTCUT MAPPING

When it comes to the command name -Keyboard shortcut (KS) mapping there are two things that designers have to consider. First the modifier(s) that they are going to use and second the hotkey. chapter 3, section 3.1.2 provides an overview about the modifiers and the hotkeys. In this section we are going to focus on the hotkey assignment

Glossary
***hotkey(s)**: the letter(s) of the KS key combination.*

We can approach the problem of improving the command name - KS mapping from different perspectives. For example we can follow the logic of existing approaches like [75, 174] which aim to create command name -KS mappings based on predefined rules. Another prespective is to view this problem from a linguistic point of view and check for semantic, lexical or phonological attributes [155] that we can exploit.

Instead we will take a more straightforward approach and focus on creating command name -KS mappings based on the letter of the command name. This is the approach that up to a point current design guidelines (chapter 3, section 3.1.2) propose [4]. Furthermore it is perhaps less complex to design command name - KS mappings focusing solely on the letters than taking other aspects of the word into account.

As a result within this thesis we are going to explore how hotkeys that have a relationship with the command's letter can impact the KS adoption. In particular we identify 3 cases:

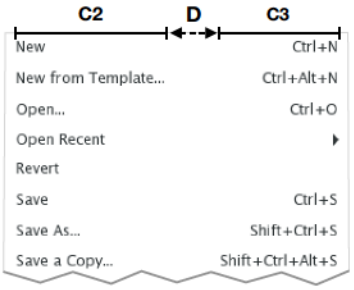
- GOOD : The letter of the KS is the first letter of the command. For example Duplicate - Ctrl + D. This level of quality inspired by the design guidelines for KS in [88, 112].
- MEDIUM: The letter of the KS is one of the letters of the command but not the first or the last letter. For example Duplicate - Ctrl + L. This level of quality was inspired by Microsoft's access keys section 3.1.2.
- BAD: The letter of the KS is not one of the letters of the command. For example Duplicate - Ctrl + G. This level of quality is inspired by current studies that evaluate interaction techniques that promote KS usage [77, 107, 108]. In these studies, experimenters use this mapping quality to make the mnemonic connection between the KS and the command name difficult to learn.

In experiment 1 section 5.3 we are going to investigate how GOOD, MEDIUM and BAD mappings affect the KS adoption.

4 In particular they suggest to use the first letter of the command whenever it is possible

5.2 REVISITING THE LINEAR MENU LAYOUT

The aim of this section is to present different alternatives of menu designs that aim to decrease the distance between the command label and the Keyboard shortcut (KS) cue. First we present the different alternatives we considered to decrease the distance between the command name and the KS cue . Then we propose a design space based on two dimensions: the position of the KS cue in relation to the command name and the alignment of the KS. Afterwards, we systematically explore this design space and generate possible menu layouts with different placements of the KS cue . Finally we choose the most promising designs to evaluate them in controlled study environment.



5.2.1 Current approaches to decrease the distance between the keyboard shortcut and command name

Several approaches can reduce the distance between command names and keyboard shortcut cues. One consists of decreasing the variability of the command name length (for more details see section 3.2, fig. 3.4 - Column 2) as the column width depends on the longest command name. In practice, this approach is not feasible as it would require to change the wording of several command names while (1) users are already familiar with many command names such as Save, Open, (2) an appropriate alternative command name does not always exist, and (3) choosing the right command name for a functionally is a challenging task for designers [13, 14].

Another approach consists of reducing the width of the third column by (1) reducing modifiers or (2) replacing modifier characters with symbols, as in Apple menus. However, since designers anyways favor simpler keyboard shortcuts, symbols do not significantly decrease the length of keyboard shortcut cues. Moreover, solutions such as reducing the overall spacing and margins, although easy to implement, save only a couple of pixels.

In contrast, our approach revisits the traditional layout regarding the *position* and the *alignment* of the menu elements. We now present this design space which we illustrate using representative examples.

5.2.2 Design space dimensions: position & alignment

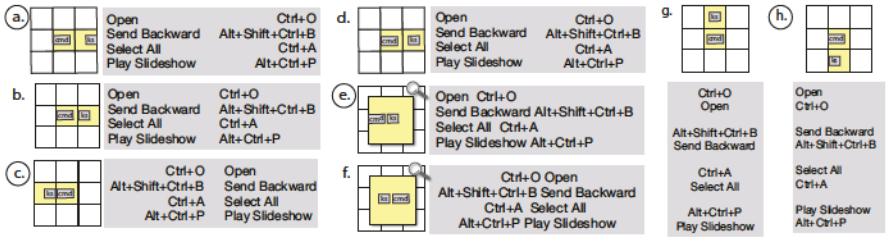


Figure 5.2: Design space for placing the keyboard shortcut cue (ks) in relation to its command name (cmd) regarding two dimensions: *position* and *alignment*. Shortcut positions: left (c,f), right (a,b,d,e) bottom (h), up (g) or diagonal (not shown in the figure). Shortcut alignments: left (b,h), right (a,c), and centered (f,g). The circled letters highlight the designs that we finally compared in Experiment 1.

The design space of the linear menu is enormous [15] as numerous variables exist for each element (e.g. font, saliency, etc.). This design space only focuses on two dimensions of the linear menu layout: the *position* and the *alignment* of the menu elements (icon, command name, keyboard shortcut cue, and submenu symbol) with the primary goal to promote keyboard shortcuts.

Representation. To describe our design space, we use an iconified representation (the grids on the left of each menu in fig. 5.2) to get a quick overview of the set of possibilities and facilitate prototyping and brainstorming among designers. The command name is represented by the icon cmd and the keyboard shortcut cue by the icon ks . Each of them is displayed in a box icon representing the whole space allocated for these elements, typically, the column width. The relative position (e.g. top, bottom, left, right) of the box illustrates the dimension *position*, while their relative position within the box illustrates their *alignment* (e.g., left , right, center). Figure 5.2 illustrates several instances of this design space by varying the position and the alignment of the two elements. A key feature of the design space is that the two elements can share the same box as shown in fig. 5.2 (e,f). It means that the designer is choosing to allocate one column for both elements. The consequence is that this pair of elements follows a unique alignment.

The same representation can be used to describe the organization of the icon and the submenu element. To minimize any possible readability impairment and simplify this analysis, we do not consider them in this section but discuss them in the section Discussion.

5.2.3 Exploration of the design space

Our objective was to identify a small subset of designs that are good test cases for the later controlled experiment. To achieve this, we first generated the 42 alternative designs (fig. 5.2 illustrates 8 of them) corresponding to the different combinations between *position* and *alignment* for both the command name and the keyboard shortcut cue. We then discarded 27 designs where the keyboard shortcut cue where located far from the command name (e.g. b-d). We then asked the opinion of 4 designers (> 6 years of experience) regarding the *readability* of the command names and keyboard shortcut cues for the 14 remained designs. This allowed us to discard 6 additional layouts (e.g., fig. 5.2-f-g) [5].

Finally, we iterated over the remaining 8 and kept the following 3 designs (seen in fig. 5.2):

- the LEFT (c) where the keyboard shortcut cues are positioned before the command names. This design was perceived as aesthetic by designers as well as familiar (i.e. similar to existing layouts such as flyers)
- the RIGHT (e) where the command names are positioned first and the keyboard shortcut cues are appended right after. Although all cumulated designs (e.g. f) received very low ratings, this one received the highest score among them on readability.
- the BELOW (h) where the keyboard shortcut cues are located below the command names; This design received the highest score among all vertical layouts (e.g. g) considering also aesthetics and familiarity.

In sections 5.4 and 5.5 we will explore how these layout can affect the KS adoption.

5.3 EXPERIMENT 1

We compared the 3 command name - Keyboard shortcut (KS) mapping qualities (GOOD, MEDIUM, BAD) on keyboard shortcut learning.

GOOD:
Duplicate - Ctrl+D.

MEDIUM:
Duplicate - Ctrl+L.

BAD:
Duplicate - Ctrl+G.

5.3.1 Participants

20 university students volunteered for this study (5 female) between 23-40 years old.

5 For details about the questionnaire that the designers had to fill out and their answers see section i.1

5.3.2 Apparatus

MENU SYSTEM ORGANIZATION

The organization of the menu system builds on the organization of the Grossman et al. [77] study. We also used 6 categories: Clothes, Office, Hobbies, Vegetables, Fruits and Animals.

Each menu category had 12 items 3 of which were target commands. For each target command and 3 non target command we assigned a KS. For the non target commands was random while for the target commands we are going to elaborate in section 5.3.3 about the assignment process. The remaining 3 commands didn't have a KS. We chose randomly which of the non target commands will have a KS and which won't. Finally, the length of the commands ranged between 6 and 10 characters.

INTERFACE

The interface is also similar to the one of Grossman et al. [77] study. The 6 menu categories were organized in a menubar which was located to the top of the screen, while the visual stimulus and the Start trial button was located on the bottom of the screen.

EQUIPMENT

We used a 23" screen, a QWERTY keyboard and a mouse. For the eye-tracking we used an tobii pro eye-tracker.

DURATION

Each participant performed the experiment in one session, lasting between 35 and 70 minutes.

5.3.3 Experimental design

TARGETS

Each category had three targets: one with Good mapping quality, one with Medium mapping quality and one with with Bad mapping quality. This resulted in 18 target items in total, 6 for each mapping quality.

TARGET FREQUENCY

We opted for a zipfian distribution for the same reasons we used the same distribution in Experiment 2. We followed the same procedure but this time for 6 items [6]. The resulted frequencies were rounded off and consisted of: 12, 6, 4, 2, 1 and 1. We used these frequencies for each mapping quality. In the beginning of each session we randomly assigned the frequencies for each item.

6 number of targets for each mapping quality

TARGET KEYBOARD SHORTCUT ASSIGNMENT

For the target items we assigned a keyboard shortcut based on the following constrictions:

- Each category will have one target with GOOD mapping quality, one with MEDIUM mapping quality and one with BAD mapping quality.
- Each letter will be used only once
- Half of the letters that we are going to use are going to be from the left side of the keyboard (Q, W, E, R, T, A, S, D, F, Z, X, C) and the the other half are going to be from the right side of the keyboard (P, O, I, U, Y, L, K, J, H, M, N, B). We took this decision to make sure that the position of the letter in the keyboard doesn't affect the user behavior.
- We randomly assigned the Ctrl modifier to half of the targets and the Shift modifier to the other half. We took this decision based on existing studies like [7, 77, 107].

MENU SYSTEMS

In all sessions the organization of the menu system was the same yet which of target commands was assigned to which command name - KS mapping varied. The reasoning behind this decision is that we wanted to ensure that it wasn't a specific command name - KS mapping that help users transition to KS but it was the mapping quality itself.

We created therefore 3 menu systems based on the restrictions that I discussed in the shortcut assignment. We rotated the mapping quality for each target in each menu system. So for example if in the MENU SYSTEM 1 a target has a GOOD command-KS mapping, in the MENU SYSTEM 2 this target will have a MEDIUM command-KS mapping, and in MENU SYSTEM 3 will have a BAD command-KS mapping. Each participant saw only one set.

STIMULUS

A pictorial stimulus appeared at the bottom of the screen that was depicting the target command.

TASK

The task was to select the target item as fast and accurately as possible by using either the dropdown menu or the corresponding keyboard shortcut [77]. In case of errors, we used the same "pop-up window" error penalty. A trial finishes as soon as participants correctly executed the command.

5.3.4 Procedure

INSTRUCTIONS
Participants were explained how to use the menubar and the KS and they were encouraged to be as fast and as accurate as possible. We did not provide any incentives to favor KS.

PERCEPTION OF PAST PERFORMANCE
At the end of each block we asked the participants to tell us what was their perception of performance during the block they just completed. In particular we asked them to tell us what was in their opinion, the average performance in terms of time while using menu, KS and average performance in general.

EYE-TRACKING AND POST-TEST During the experiment we used an eye tracker to see where where users' visual attention was during the experiment. We calibrated the eye tracker before the beginning of the experiment and we made sure that the users were approximately 65cm away from the screen. We then instructed them to keep this distance as steady as possible.

At the end of the experiment we asked the participants to fill out a questionnaire. We first asked them questions regarding they gender and age and then we asked to indicated how often they use KS in their everyday life. Finally we showed them the visual stimulus for each command and asked them to indicate the corresponding KS, how confident they were for their answer and if they had a specific strategy to memorize the KS.

In this experiment we are not going to report this data. We are going to do that in a future work.

5.3.5 Design

A mixed design was used. The between-participants independent variable was the target-KS mapping set. Each participant was randomly assigned to one of the three sets. The within-participant variable was the item frequency. For each mapping quality category we assigned a different frequency for each item. The item to frequency mapping was counterbalanced across all participants, with each item mapped to each frequency equal amount of times. Within each block, items were presented randomly. The number of appearances for each item matched the assigned frequency of the zipfian distribution. Overall, the design was 20 participants x 10 blocks x 78 trials = 15600 selections.

5.3.6 Results

5.3.6.1 Keyboard shortcuts adoption

KEYBOARD SHORTCUTS PER MENU SYSTEM

We first investigated whether a specific command-Keyboard shortcut (KS) mapping affected the user's behavior. Figure 5.3 reports the results for (A) the mean rate of *correct keyboard shortcuts* between the command-KS mapping set and the (B) the mean rate difference between the sets

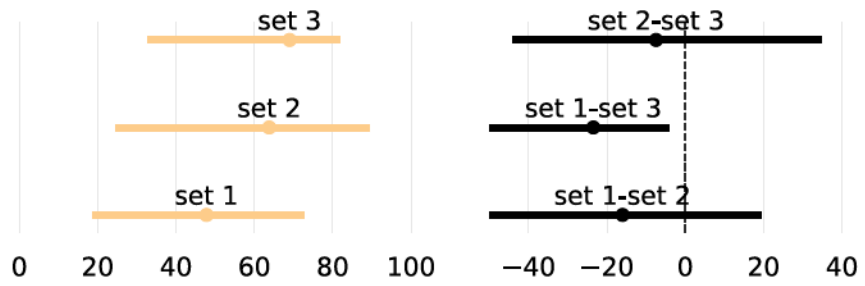


Figure 5.3: *(A)*: Mean rate of *correct keyboard shortcuts* per mapping quality. *(B)*: Mean rate difference of *correct keyboard shortcuts* between command-keyboard shortcut mapping. All error bars are 95% CIs.

We didn't observe any significant differences between the menu sets with the exception of some small effect between the MS1 and MS3 [7]. We do observe though that all the CIs are wide which may indicate that the number of participants wasn't big enough

KEYBOARD SHORTCUTS ADOPTION PER MAPPING QUALITY

Figure 5.4 shows both (A) the mean rate of *correct keyboard shortcuts* for each mapping quality and (B) the mean rate difference of *correct keyboard shortcuts* between the mapping qualities. For each proportion we report the 95% CI indicating its range of plausible values.

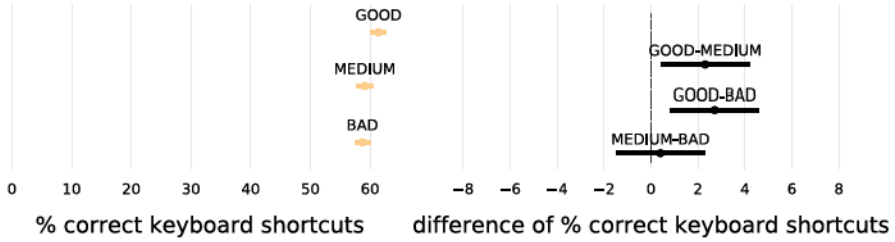


Figure 5.4: *(A)*: Mean rate of *correct keyboard shortcuts* per mapping quality. *(B)*: Mean rate difference of *correct keyboard shortcuts* between mapping qualities. All error bars are 95% CIs.

7 We consider this effect small due to the width of the CI and how close is to 0

Results suggest that the GOOD mapping has a small advantage over the MEDIUM and BAD. These differences remain the same even if have $Included_{>0}$ and $Included_{>25\%}$ but the mean rate values have been shifted to the right.

Since we saw that the menu systems affected the users' behavior we analyzed also our results for each set. Figure 5.5 reports the mean rate difference between each mapping quality for each set. We observe that for set 1 the GOOD out-performed the other two conditions. For the other the two sets though results are inconclusive.

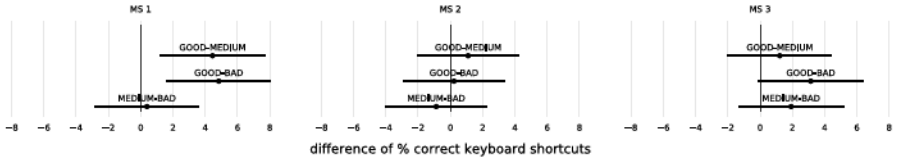


Figure 5.5: Mean rate difference of *correct keyboard shortcuts* between mapping qualities for each command-keyboard shortcut mapping. All error bars are 95% CIs.

KEYBOARD SHORTCUT ADOPTION PER FREQUENCY

Similarly with Experiment 2 the results of the frequency analysis showed that mean rate *correct keyboard shortcuts* was higher for higher frequencies. For more details see chapter i.

5.3.6.2 *Errors and time*

For the error we performed two kind of analyses. First we performed analyses for each mapping quality to see the mean number of errors. Results were inconclusive and therefore we cannot draw any conclusions from this analysis.

Then we performed an error analysis for only the trials that users selected the command using KS. We distinguished 3 kind of errors: 1. $Error_{letter}$: users selected the correct modifier but the wrong letter, 2. $Error_{modifier}$: users selected the wrong modifier but the correct letter, 3. $Error_{modifier_and_letter}$: users selected the wrong modifier and the wrong letter. For each of the above errors we performed an analysis per mapping quality to see the average number of errors.

Figures 5.6 and 5.7 report the results of this analysis (for an analysis per menu system see section i.2). As a general remark for all the cases users made more errors on the modifier than the other cases. Examining each error type separately we observe that for the case of $Error_{letter}$ results were inconclusive which may indicate that users could remember equally well all the letters. In the case of $Error_{modifier}$ the BAD seems to have a small advantage over the other two conditions while in the case of $Error_{modifier_and_letter}$ the GOOD seems to have a small advantage over the other two conditions.

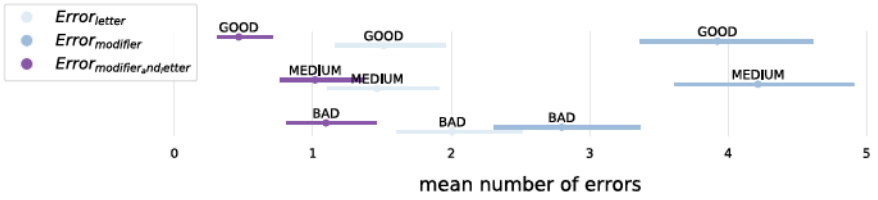


Figure 5.6: Mean number of errors per layout for each error type. All error bars are 95% CIs.

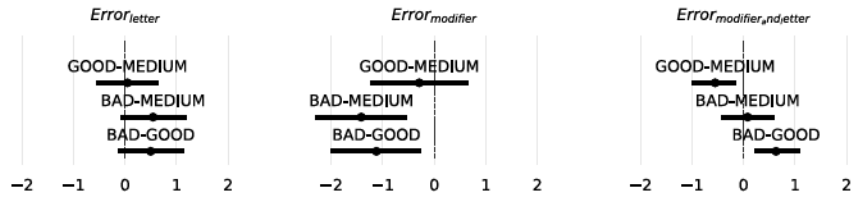


Figure 5.7: Mean number of errors difference per layout for each error type. All error bars are 95% CIs.

We conducted also a CI analysis for trial completion time for each mapping quality but no significant effect was observed.

5.3.6.3 *Perception of past performance*

Figure 5.8 reports the results of our analysis for the user's perception of performance for both menus and KS for each block. We observe that on average subjects believed from the end of the first block that they were faster with KS than with menu.

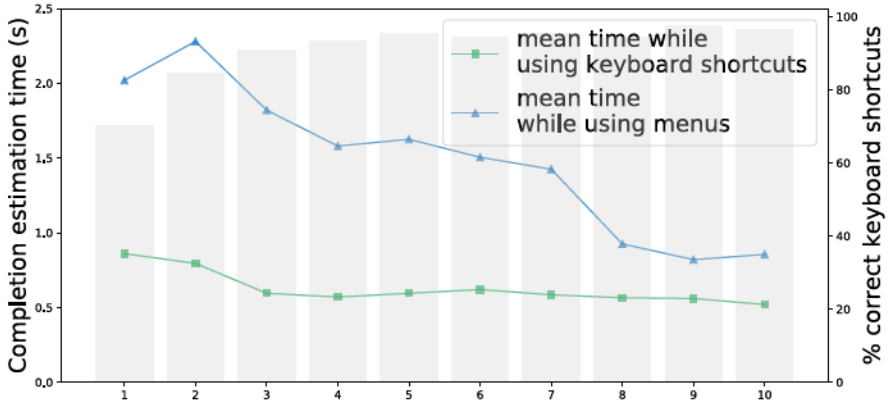


Figure 5.8: Mean perception of performance (in seconds) for menu and keyboard shortcut for each block. In the background we also show the % mean rate of *correct keyboard shortcuts* per block

5.3.7 Discussion

Findings from this experiment indicate that the GOOD mapping quality promotes KS usage. It is not clear if there is a difference between the

other two conditions though. One aspect we should consider is that we may needed more participants for this study given the results of fig. 5.3.

5.4 EXPERIMENT 2

We compared the 3 selected layouts (LEFT, RIGHT, and BELOW) to the traditional linear menu layout (BASELINE) on keyboard shortcut learning.

5.4.1 Participants

We hired 27 University students (20-30 years old) [8]. Their payment was 10 €/hour. The 3 fastest participants received an extra bonus of 20,15 and 10 € respectively.

5.4.2 Apparatus

MENU SYSTEM

Our experiment design builds on Grossman's et al. study [77] (for more details see chapter 3, section 3.3). However we made several changes for reasons we now detail.

First we use 4 instead of 6 menu categories in our menu system. The reason for this change is that we wanted to evaluate 4 different menu layouts (BASELINE, LEFT, RIGHT and BELOW). Therefore in each session each menu was assigned each layout (more details in the experimental design section that follows). The 4 menu categories were : Animals, Fruits, Office and Vegetables.

Another change from Grossman et al. study is the length of the target command names. In their experiments the command names were limited from 3 to 10 characters. In our experiment though we wanted to create menus that had a "realistic" variety of command lengths. a database of about 30 frequent Mac OS applications (1048 menus in total) [14] and we computed the mean command name length (mean= 10 characters) and the variance [33] (24, with a 95% bootstrap confidence interval of [22, 25]). From this database, we randomly pick one 12-item menu with command name length distribution: 4-5-6-7-12-13-13-13-14-15-17-18.

Our command names were common expressions. Some of them contained 2-3 words (e.g., red fish) to precisely control their length (i.e. the total number of characters).

Finally similar to Grossman et al. we used a BAD command name - Keyboard shortcut (KS) mapping.

8 The planned size was 28, but we lost the data of one participant due a technical crush.

INTERFACE

The interface is similar to the one of experiment 1. The 4 menu categories were organized in a menubar which was located to the top of the screen, while the visual stimulus and the Start trial button was located on the bottom of the screen.

EQUIPMENT

We used a 23" screen, a QWERTY keyboard and a mouse. For the eyetracking we used an tobii pro eyetracker.

DURATION

The experiment lasted 1 hour on average.

5.4.3 Experimental design

TARGETS

To use the command name length as a factor in this experiment, each of the 4 menus contained 3 targets: the SMALL word (4 characters), the MEDIUM word (13 characters), and the LONG word (18 characters). This resulted in 12 target items in total (instead of 14 in [77]).

TARGET FREQUENCY

We used an uniform frequency distribution (all items have the same frequency) rather than a zipfian distribution [77] (some items are more frequent than the others), as we were interested in comparing both technique and individual item performance. Using a zipfian distribution could bias our results [103] or result in a complex experimental design.

STIMULUS

The visual stimulus was an image depicting the target command (e.g. a "red fish").

TASK

The task was to select the target item as fast and accurately as possible by using either the dropdown menu or the corresponding keyboard shortcut [77]. For wrong selections, a pop-up window appeared on the center of the screen. As a penalty, participants had to close this window and perform the correct selection (in [77] used 3 sec delay penalty instead). A trial finishes as soon as participants correctly executed the command.

5.4.4 Procedure

PRE-TEST

We first asked participants to execute a sequence of keyboard shortcuts. The reason was that in previous studies, we observed that some participants did not know what is a keyboard shortcut.

INSTRUCTIONS

Participants were explained how to use the menubar and the keyboard shortcuts and were encouraged to be as fast and accurate as possible. We strongly emphasized that they are free to use any method they want. We then displayed an observation that some previous studies indicate that using shortcuts can be faster. We also noted that it is acceptable to do some mistakes in the beginning to motivate risk-averse persons to use keyboard shortcuts.

SELECTION TEST

As described in section 5.4.3, participants had to identify the target stimulus to execute its corresponding command. Once the command was successfully executed, they could proceed to the next trial.

EYE TRACKER AND POST TEST

We again used an eye tracker. We followed the same steps to calibrate as with experiment 1.

Like with experiment 1, at the end of the experiment we asked the participants to fill out the same questionnaire. The questionnaire consisted of questions about demographics (age, gender), previous familiarity with KS and questions to see if they learned the KS that they used.

In addition, the attention and memorization abilities of our participants are tested post-experiment using the Trail Making Test [10] for visual attention and task switching abilities and the Short Memory Test [116] for the short term memory and the working memory.

In this experiment we are not going to report this data. We are going to do that in a future work.

5.4.4.1 Design.

Participants were assigned to all *layouts* (BASELINE VS. RIGHT VS. BELOW VS. LEFT) following a within-subjects design. For each condition, they performed 45 blocks of 12 selections. Each block contained 3 groups of *command lengths* (SMALL, MEDIUM and LONG). Each target appeared once. Overall, the design was 27 participants × 45 blocks × 12 trials (3 command lengths × 4 targets) = 14580 selections.

5.4.5 Results

5.4.5.1 *Keyboard shortcut adoption per layout*

Keyboard shortcut adoption is generally measured as the mean rate of *correct keyboard shortcuts*, i.e the proportion of trials where participants successfully used the shortcuts without any aid or making any error. fig. 5.9 shows both (A) the mean rate of *correct keyboard shortcuts* for each layout and (B) the mean rate difference of *correct keyboard shortcuts* between the layouts[9]. For each proportion, we report the 95% confidence interval (CI) indicating its range of plausible values.

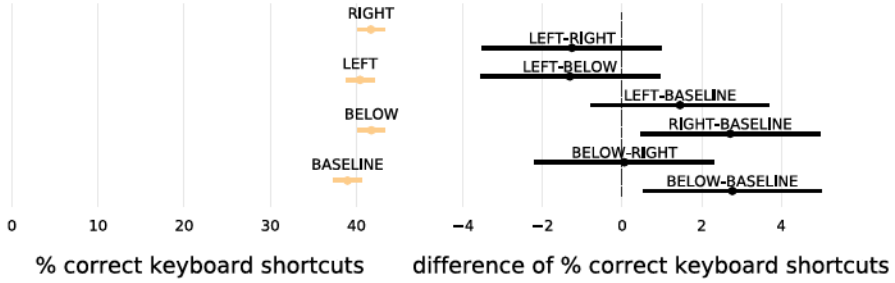


Figure 5.9: *(A)*: Mean rate of *correct keyboard shortcuts* per layout. *(B)*: Mean rate difference of *correct keyboard shortcuts* between layouts. All error bars are 95% CIs.

Results suggest that the mean rate difference of *correct keyboard shortcuts* was inconclusive between most layouts. However, we observed a small advantage of the BELOW and the RIGHT over the BASELINE, and a weaker evidence for the LEFT layout.

5.4.5.2 *Keyboard shortcut adoption per command length*

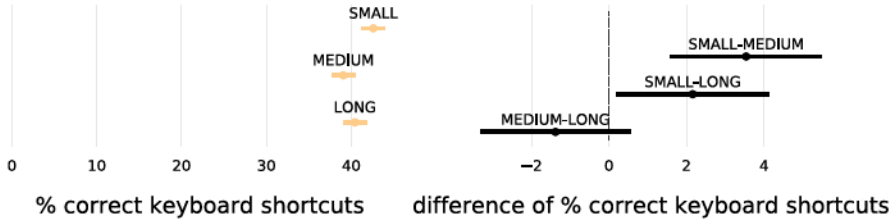


Figure 5.10: *(A)*: Mean rate of *correct keyboard shortcuts* per command length. *(B)*: Mean rate difference of *correct keyboard shortcuts* between command lengths. All error bars are 95% CIs.

We perform the same analysis for each command length (SMALL, MEDIUM, LONG). Figure 5.10 both (A) the mean rate of *correct keyboard shortcuts* for each command length and (B) the mean rate difference of *correct keyboard shortcuts* between the command lengths. For each proportion, we

9 Positive values indicate that the layout on the left outperforms the one on the right.

report the 95% confidence interval (CI) indicating its range of plausible values.

Results indicate that there is an effect between the SMALL and the other two conditions. We performed the same analysis for each command length (SMALL, MEDIUM, LONG) for each layout individually to see if one or more of the layouts are responsible for this effect. The results from this analysis showed that when participants were exposed to the LEFT and BELOW layouts, the length of the command did not seem to affect shortcut adoption. In contrast, for BASELINE and RIGHT, MEDIUM commands under-performed in comparison to the other lengths.Figures 5.11 and 5.12 show the results of this analysis for the BASELINE and RIGHT respectively [10].

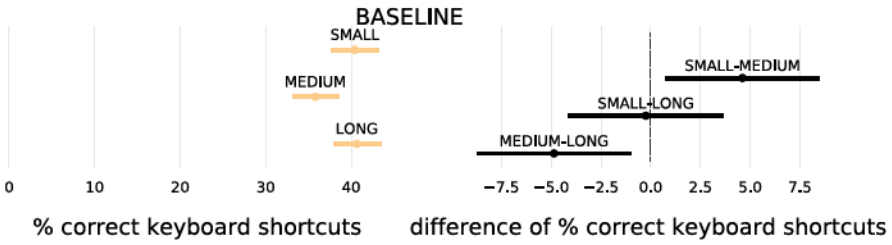


Figure 5.11: Mean rate of *correct keyboard shortcuts* per command item length for the BASELINE menu layout. All error bars are 95% CIs.

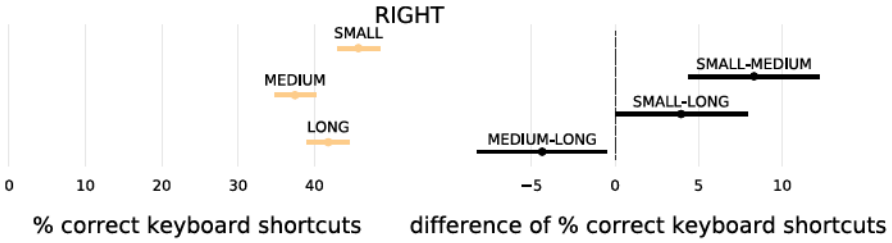


Figure 5.12: Mean rate of *correct keyboard shortcuts* per command item length for the RIGHT menu layout. All error bars are 95% CIs.

5.4.5.3 *Errors and Time*

. We conducted CI analysis for both error and trial completion time for each layout and command length, but no significant effect was observed.

5.4.5.4 *Participant inclusion criteria for data-analysis*

In similar studies [77, 97] a common practice is to re-analyze the data by including participants who did meet a specific criteria. Usually this criterion is an inclusion threshold of *correct keyboard shortcuts* and only

10 for all the results see section i.3

participants above this threshold are included to the analysis. The reason for doing this is to examine how the various interaction techniques affect the behavior of the users who tried to use *correct keyboard shortcuts*.

We followed the same strategy to examine whether the menu layouts affect the users who tried to use *correct keyboard shortcuts*. We define three levels of inclusion criteria based on current literature:

- $Included_{>=0}$ which represents all participants whether they used *correct keyboard shortcuts* or not [11]
- $Included_{>0}$ which represents the participants who tried to use *correct keyboard shortcuts* at least once. We reuse this threshold from papers like [77] which have used the same threshold
- $Included_{>25\%}$ which represents users who have used *correct keyboard shortcuts* for at least 25% of the total trials. We reuse this threshold from [97] and it aims to capture participants who at least had some experience with *correct keyboard shortcuts*.

RE-ANALYSIS FOR EACH LAYOUT

Figures 5.13 and 5.14 show the result of this analysis. Figure 5.13 shows the mean rate of *correct keyboard shortcuts* for each layout and each threshold. Figure 5.14 shows the mean rate difference between the layouts for each inclusion criteria.

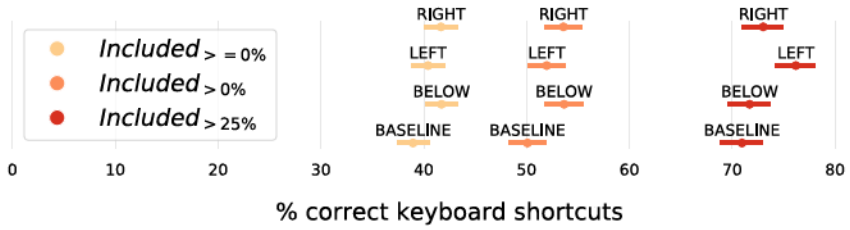


Figure 5.13: Mean rate of *correct keyboard shortcuts* per layout for each threshold. All error bars are 95% CIs.

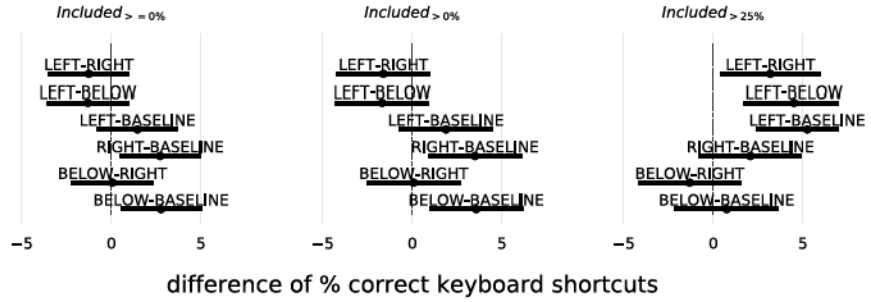


Figure 5.14: Mean rate difference of *correct keyboard shortcuts* between layouts for each threshold. All error bars are 95% CIs.

11 this is the threshold we have used in our analysis so far

Changing the threshold from $Included_{>=0}$ to $Included_{>0}$ shifted the *correct keyboard shortcuts* rates to the right ($\approx$ 10%) but the trend of menu layouts remains similar. Indeed if we observe the mean rate difference between the layouts, we observe again a small advantage of the BELOW and the RIGHT over the BASELINE, and a weaker evidence for the LEFT layout.

Changing the threshold from $Included_{>=0}$ to $Included_{>25\%}$ shifts again the values to the right but this time it seems that LEFT has an advantage over the other layouts. In particular there is an advantage of LEFT over BELOW and BASELINE and a smaller advantage over RIGHT. It is noteworthy that results for the difference between the other layouts are inconclusive in contrast with the previous thresholds where BELOW and RIGHT had an advantage over BASELINE.

RE-ANALYSIS FOR EACH COMMAND NAME LENGTH

Figures 5.15 and 5.16

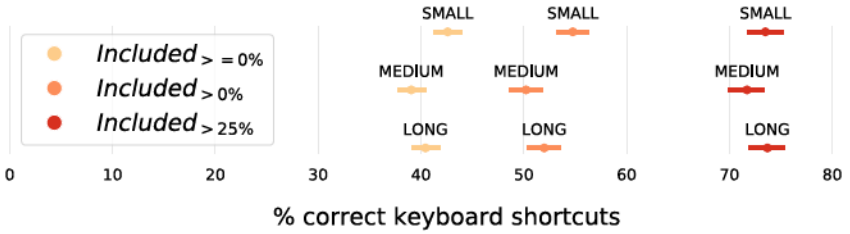


Figure 5.15: Mean rate of *correct keyboard shortcuts* per command name lengths for each threshold. All error bars are 95% CIs.

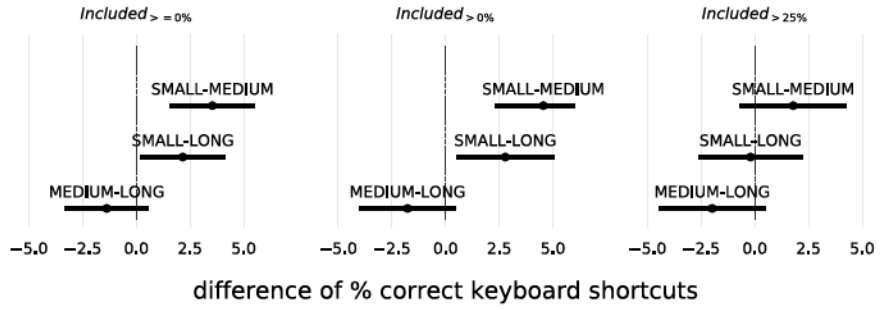


Figure 5.16: Mean rate difference of *correct keyboard shortcuts* between command name lengths for each threshold. All error bars are 95% CIs.

5.4.6 Discussion

Although an initial analysis of the layouts indicate inconclusive results with a small preference over RIGHT and BELOW layouts when we consider participants who had tried to use *correct keyboard shortcuts* many times ($Included_{>25\%}$) the LEFT layout seems to perform slightly better

over the other layouts. One possible explanation for these results is that the placement of the elements affect the participants that have already started the transition. However, our results regarding the command length do not fully explain the difference between layouts.

To explain our finding, we need to understand how users do visual search with respect to the role of command length. One visual search strategy consists of reading the command names (from left to right). With experience, users can skim only the first characters (or words) to find the desired command [31]. Another strategy is to use the salient features of the command name. For instance, several models of menu performance use the length of an item as a salient feature [34, 101]. Users can thus focus on the end of the command name, as a way to filter items by their length.

Our layouts, LEFT and BELOW, always display the cues close to the first characters of the command name. Given the good performance of these layouts and their lack of difference among commands lengths, it suggests that users mainly adopt the first visual search strategy, i.e. primarily focus on the first characters.

The analysis of the RIGHT layout is also consistent as the position of the cues is not systematically close to the first characters of the command name. Instead, the position of the cue depends on the length of the command. fig. 5.12 suggests that SMALL lengths favor keyboard shortcut learning, which is consistent with our analysis, as the SMALL is the only condition where the cues are always close to the first characters of the command name.

In contrast, the analysis of the BASELINE layout suggests that the keyboard shortcut cues should be located *close to the last characters* of the command names. This layout is interesting because the length does not influence the distance of the keyboard shortcut cues from the first characters, but only from the last ones. For the BASELINE, fig. 5.11 suggests that long command names favor keyboard shortcut learning, suggesting that the users adopt the second visual search strategy, i.e. focusing on the last characters of the command name to find the target. This contradicts our first analysis.

Finally regarding the effect of *spatial proximity* it seems that it can benefit the KS adoption. The result from our analysis indicate that LEFT especially favors KS.

5.5 EXPERIMENT 3

In this experiment, we propose a novel experimental design to favor ecological validity regarding the instructions, the methods, the interface, the distribution frequency, and the task. We further reduce the risk of having too many participants that do not transition by introduc-

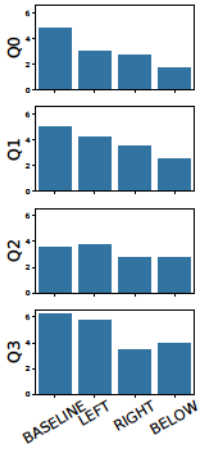


Figure 5.17

ing safeguards before (recruitment) and during the experiment (using usage tips) to ensure having enough data to analyze.

The main goal of this experiment is to investigate the advantages of LEFT in comparison to the BASELINE layout. We decided to to choose the LEFT layout for two reasons. First based on the responses of the designers (for a brief view see section 5.5 for more details see section i.1) it seems that LEFT layout is more promising than BELOW and RIGHT especially when it comes to Q3:"The user will notice the keyboard shortcut of each command in this menu". The second reason is that the reanalysis of the results using the inclusion thresholds for each menu layout (fig. 5.13) indicates that for $Included_{>} = 25\%$ threshold the LEFT performed better.

5.5.1 Participants

We recruited 42, different from Experiment 1, university students (20-30 years) from different fields (e.g. engineering, law, medicine). Payment was identical to Experiment 1. We adopted a recruitment protocol similar to Lafreniere et al. [97] to ensure that enough participants used the shortcuts. Prior to data collection, we decided to keep recruiting participants until at least 15 of them in total (regardless the condition), used *correct keyboard shortcuts* (i.e. no aid/error) for at least 25% of the trials ($Included_{>25\%}$).

5.5.2 Apparatus

Methods to execute commands: We reused menubar and keyboard shortcuts from the Experiment 1, except that, depending on the condition, *all* submenus now have the same layout (all BASELINE or all LEFT). Moreover, we added the context menu as a common method to select commands [12]. The context menu had identical hierarchy and layout to the menubar (see fig. 5.19). Participants were thus free to use the menubar, the context menu or the keyboard shortcuts to execute commands.

12 A context menu is activated with the right button of the mouse on an object of interest (e.g. a shape in PowerPoint, a selected text in Microsoft Word) or directly on the background of the application. The menu content depends on the object of interest or replicates the content of the menubar, e.g., software GIMP.

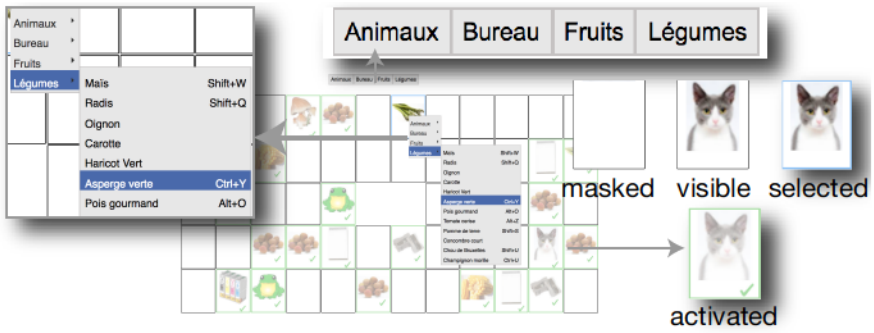


Figure 5.19: (A) Overview of the new interface and (B) the 4 stages for each target item.

Interface. The new interface (fig. 5.19), displayed a 5x12 grid in the center of the screen. Each cell had four states:

- *masked*: No image is shown.
- *visible*: An image indicates the target command, i.e. the command that the user has to execute on this cell.
- *selected*: The image is outlined in blue once participants click on it (with the left or right mouse button). Then they can use their preferred method to select the command (menubar, context menu, or shortcuts).
- *activated*: The image is blurred, checked and outlined in green to indicate a successfully selected command.

5.5.3 Experimental design

5.5.3.1 *Frequency*

In contrast to Experiment 1 (uniform distribution), we used a zipfian distribution for the frequency of appearance of each target to reflect real application usage [179]. We used the standard zipfian distribution equation [103, 190] (exponent==1) and applied it to the set of the 12 target commands (see Experiment 1) following the procedure described in [77]. The resulted frequencies were rounded off and consisted of: 12, 12, 6, 6, 4, 4, 3, 3, 2, 2, 2, 2. In the beginning of each session each item was randomly assigned a frequency.

5.5.3.2 *Stimulus & Task*

The targets were identical to the Experiment 1, but they now appeared in a cell inside the grid interface (fig. 5.19). Participants had to select a cell before executing the command and then select the next cell, until all cells are activated.

The rational behind this design choice was to use a high level task ("complete the grid") rather than series of low-level tasks (execute individual commands). Forcing participants to interact with multiple objects-of-interest (cells) located at different positions on the screen can potentially foster more realistic mouse behavior. Moreover, it may help participants to better perceive the "real cost" [48] of mouse-based commands. Indeed, the "real cost" of menu-based methods [48] includes not only the time to reach the menu widget (e.g. menubar) and the time to navigate in the menu system, but also the time to *return back* to the objects-of-interest [48] (refer to [13] for extended discussion). Although most studies usually measure only the time that users need to execute a command (e.g. [7, 77, 107, 108, 133]), participants might neglect the time to return back to the objects-of-interest and thus over-estimate the performance of menu-based techniques in comparison with keyboard shortcuts.

5.5.4 Procedure

PRE-TEST

As in experiment 2, we asked participants to press sequences of keyboard buttons. However, unlike before, they were not connected to a screen, we asked them to also press mouse buttons, while the keys did not correspond to known keyboard shortcut combinations.

INSTRUCTIONS

In this experiment, participants were told to complete the grid by executing the corresponding commands on each image, but they were not informed about the available methods. So, participants did not receive incentives favoring keyboard shortcuts.

While this scenario is more realistic, the risk is that a large proportion of participants may not try to use keyboard shortcuts (lack of awareness, lack of motivation, etc.). It thus introduces two major drawbacks.

First, much more participants are needed to observe enough keyboard shortcuts adoptions. For example, in [77] 50% of participants did not transition to shortcuts for the BASELINE, while in our pilots, involving non-computer scientist participants, the no-transition percentage was even lower (75%). In Experiment 1, approximately 33% participants did not transition even though in the instructions we informed them about the benefits of the keyboard shortcuts. To avoid having too many participants that do not transition we introduced tips.

Second, the data from participants who did not transition are difficult to reuse as our focus is on the transition phase. To filter out these participants we defined an inclusion threshold.

We are going to expand more on both the tips and the inclusion thresholds in the rest of this section.

TIPS

From pilots and Experiment 1 data analysis, we saw that participants are very likely to "never" transition, if they do not transition during the first three blocks. For this reason, after the third block, we added tips during the inter blocks (such as the ones we can find in modern applications, e.g. Pycharm [130]) to encourage the use of keyboard shortcuts. We added 3 tips appearing at three different timestamps. The first tip (inter-block 5) informed participants that there are three available methods to execute commands (menubar, context menu and keyboard shortcuts). The second tip (inter-block 6) informed them that studies have shown that keyboard shortcuts are faster than menus. The third tip (inter-block 7) explicitly informed them that they should use keyboard shortcuts to optimize their performance. Once a tip was shown, it remained visible during all the following inter-blocks.

EYE TRACKER AND POST TEST

We again used an eye tracker. We followed the same steps to calibrate as with the previous experiments.

After the test we also followed the same procedure as with experiment 2. First we asked the participants to fill out the same questionnaire about demographics (age, gender), previous familiarity with KS and questions to see if they learned the Keyboard shortcut (KS) that they used. Then we asked participants to perform a Trail Making Test [10] and a Short Memory Test [116]

In this experiment we are going to report the questionnaire results for previous familiarity with KS.

DESIGN

The experiment had a mixed-subjects design. Each participant was randomly assigned to one *layout* (BASELINE VS. LEFT) to avoid possible learning transfer from one layout to another. The within design factor was the frequency of the commands. The position of targets in the grid was randomly assigned in each block. Each participant performed 10 blocks of 58 selections. Overall, the design was 2 layouts × 21 participants × 10 blocks × 58 trials = 24360 selections.

INCLUSION THRESHOLD

Based on the analysis of Experiment 1, we considered only participants who used *correct keyboard shortcuts* (i.e. no aid/error) for at least 25% ($Included_{>25\%}$) threshold of the trials, by excluding 7 participants. We took this decision before starting collecting the data and

5.5.5 Results

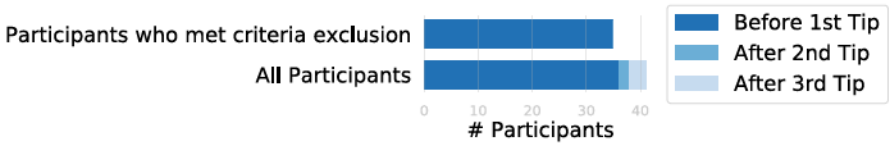


Figure 5.20: % Participants who started the transition before the first tip and after the first, second and third tip respectively.

fig. 5.20 shows when participants started the transition (i.e., used *correct keyboard shortcuts* for the first time) with respect to the different tips. The first bar shows that all the participants included in the analysis started transitioning before they saw the first tip. The second bar shows the same information, if we add the 7 excluded participants. Although we used the same recruitment methods we used in experiment 1, Overall in experiment 2 participants used more keyboard shortcuts . To compare, in Experiment 2 we had to exclude 16% of the participants while in Experiment 1 we excluded 33%.

5.5.5.1 *Keyboard shortcut adoption*

To measure the keyboard shortcut adoption, we used identical methodology and metrics to the Experiment 1.

KEYBOARD SHORTCUT ADOPTION PER LAYOUT

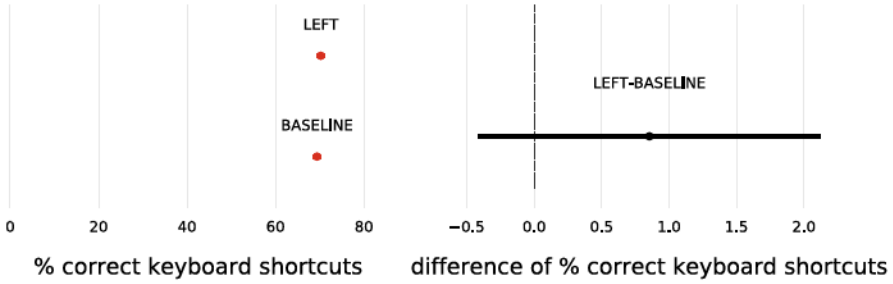


Figure 5.21: *(A)*: Mean rate of *correct keyboard shortcuts* per layout. *(B)*: Mean rate difference of *correct keyboard shortcuts* between layouts. All error bars are 95% CIs.

Figure 5.9 shows both (A) the mean rate of *correct keyboard shortcuts* for each layout and (B) the mean rate difference of *correct keyboard shortcuts* between the layouts[13]. Again, for each proportion, we report the 95% confidence interval (CI) indicating its range of plausible values. We observe that, unlike in the Experiment 1, the mean rate difference between the BASELINE and LEFT layout remains inconclusive.

13 Positive values indicate that the layout on the left outperforms the one on the right.

KEYBOARD SHORTCUT ADOPTION PER COMMAND LENGTH

We performed the same analysis for each command length (SMALL, MEDIUM, LONG) and for each layout (LEFT, BASELINE) individually. The results for LEFT (fig. 5.23) are generally consistent with the results of Experiment 1 although this time it seems that there is a small advantage of LONG to MEDIUM. In contrast, with the BASELINE layout the SMALL command under-performed in comparison to the other lengths (fig. 5.22).

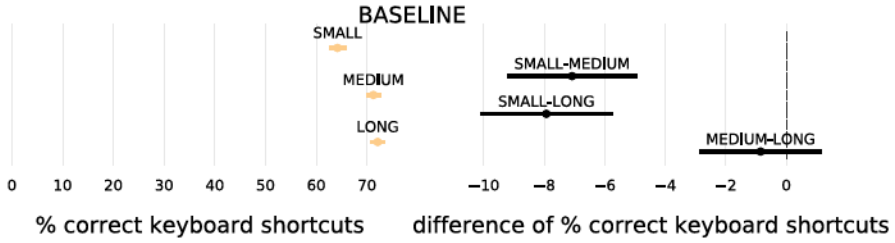


Figure 5.22: Mean rate of *correct keyboard shortcuts* per command item length for the BASELINE menu layout. All error bars are 95% CIs.

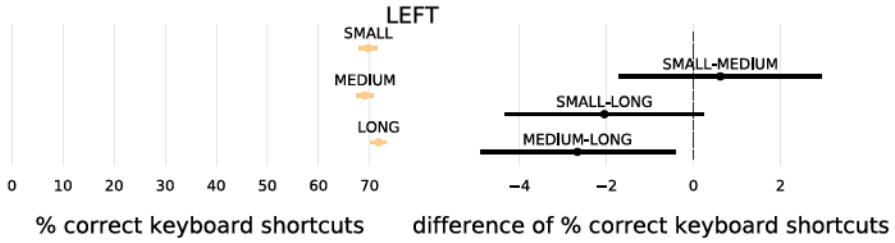


Figure 5.23: Mean rate of *correct keyboard shortcuts* per command item length for the RIGHT menu layout. All error bars are 95% CIs.

KEYBOARD SHORTCUT ADOPTION PER FREQUENCY

We conducted a similar analysis for each frequency for all the data and for each menu type individually. In all cases the results are consistent with other studies that are using zipfian distribution e.g. [77]. The higher the frequency the higher the mean rate of *correct keyboard shortcuts*. For more details see chapter i.

5.5.5.2 *Keyboard shortcut adoption per user profile*

In the Experiment 1, the results were conclusively in favor of the LEFT layout for keyboard shortcut learning for $Included_{>25\%}$. In contrast, in this experiment, the two layouts turned out to be hard to distinguish. The effects are likely small requiring perhaps larger statistical power to be estimated reliably. Therefore, we decided to examine if the profile of the users can be a useful differentiating factor.

The analysis in this section has not been specified before seeing the data and is therefore post-hoc analysis.

In both experiments, participants reported how often they use keyboard shortcuts in their daily life, dividing them into FAMILIAR and NOT-FAMILIAR users. fig. 5.24 shows that most Experiment 1 participants (13 out of 19) were NOT-FAMILIAR with shortcuts. In contrast, the Experiment 2 ones were equally spread between the two groups.

Figure 5.24: # of participants who were FAMILIAR and NOT-FAMILIAR with keyboard shortcuts for both experiments.

To investigate whether familiarity played a role in our results, we re-analyzed the data for each familiarity group.

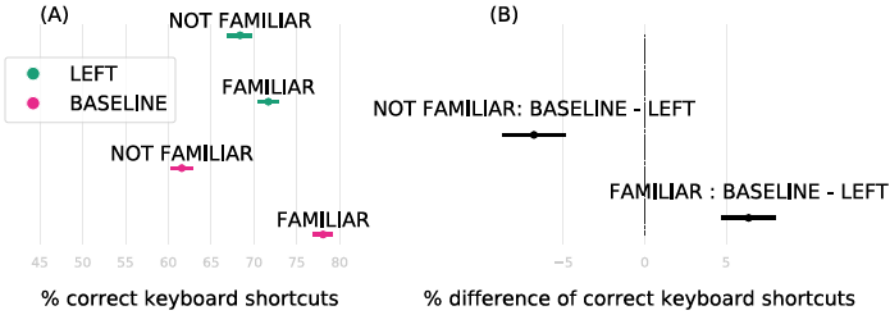


Figure 5.25: *(A)*: Mean rate of *correct keyboard shortcuts*. *(B)*: Mean rate difference of *correct keyboard shortcuts*. Black CIs indicate differences between the two layouts for each user profile (FAMILIAR and NOT-FAMILIAR).

Figure 5.25 reports the results concerning the mean rate of *correct keyboard shortcuts* for each familiarity group. We observe that the LEFT layout seems more appropriate for novice users, whereas the BASELINE seems more appropriate for users who are already familiar with keyboard shortcuts usage.

5.5.6 Discussion

Based on the findings of the Experiment 2, we expected to see a difference between LEFT and BASELINE in the Experiment 2. Surprisingly, the results of the Experiment 2 were overall inconclusive regarding which of the two layouts can be more effective for keyboard shortcut adoption. We discuss possible explanations why the two experiments differed in their findings.

There are several differences between the two experiments regarding the task and the instructions. However, it is unclear to us how the task could have influenced the relative performance of the two layouts.

Instructions were more likely to be an explanation. By providing less incentives, participants should be more likely to use less keyboard shortcuts, as reported in previous experiments [77]. However, we observed the opposite effect. 13% more participants used *correct keyboard shortcuts* in experiment 2 than in experiment 1.

One explanation for the usage of Keyboard shortcut (KS) is that the participants' expertise differs in the two experiments. Indeed as we saw in the post questionnaires (see fig. 5.24 in experiment 2 approximately 32% of the users were already familiar with KS while in experiment 3 this percentage was approximately 52%. Therefore we performed a post-hoc analysis to see how previous familiarity with KS affected the results of our experiment.

The results of this post-hoc analysis provided some interesting insights as we saw a difference between the BASELINE and LEFT layouts for each familiarity.

One explanation for these results is that prior user familiarity with the keyboard shortcuts influenced the relative performance of each layout: NOT-FAMILIAR users benefit from using LEFT, while the experience of users (FAMILIAR group) with the traditional linear menu impaired their use with the LEFT, likely due to the cost of changing habits. Another explanation is that LEFT is not sensitive to previous familiarity with KS since is a novel design

Another explanation is that LEFT layout is not sensitive to previous familiarity because it is a novel design for all the participants (regardless expertise and familiarity). In contrast BASELINE is more sensitive to expertise. Indeed a naked-eye analysis of fig. 5.25-A indicates that for LEFT the means difference between the FAMILIAR and NOT-FAMILIAR users for LEFT is small while the mean difference for BASELINE is bigger.

To conclude further studies that have a more careful recruitment method regarding familiarity with KS can shed more light about the potential benefits of LEFT. We still believe that decreasing the distance between the command label and KS cue can favor KS adoption.

5.6 CONCLUSION

To summarize this chapter we discussed the spatial and semantic relation between the command and Keyboard shortcut (KS). Results from our experiments 2 and 3 point to the direction that spatial proximity and easy to remember KS can help shortcut adoption. While results from experiment 1 indicates that especially GOOD mapping can promote KS adoption.

One problem that we encountered in this chapter is how to analyze our data properly. In order to gain better insights we used various thresholds in order to focus on different types of users. But the thresholds we chose may seem arbitrary and it is quite possible that choosing

different thresholds would yielded different results. In the next chapter I am going to discuss more about this.

Finally so far our analysis doesn't provide enough insights about the menu to KS transition. In this chapter (as well as chapter 4) we analyzed our data following existing methodologies focusing on the *whole* KS adoption. Yet in chapter 2 I discussed theories and models that can offer more insights about the transition.

This type of analysis is not trivial though and to the best of our knowledge there isn't any established methodology to do just this. The goal of the next part is to provide a data analysis methodology that can provide more insights about the transition.

Part III

CHARACTERIZING THE MENU TO KEYBOARD SHORTCUT TRANSITION

6 CHARACTERIZING MENU TO KEYBOARD SHORTCUT: TRANSITION START, TRANSITION DURATION AND PERFORMANCE

Gilles Bailly, Emmanouil Giannisakis, Marion Morel, Catherine Achard. Caractériser la transition des menus vers les raccourcis claviers. AFIHM. 30eme conférence francophone sur l'interaction homme-machine, Oct 2018, Brest, France. IHM-2018, pp.30-41, 2018, Articles Scientifiques.

Chapter 5 included three studies investigating the impact of different factors (e.g. menu layout) on the transition from menus to keyboard shortcuts. The main measure was the percentage of correct keyboard shortcuts and is used in several studies [7, 77, 107, 108, 133, 143, 189]. This measure informs about the average usage of correct Keyboard shortcut (KS), but not about the transition itself. In contrast, chapter 2 presented theories and models from cognitive sciences to describe such transitions.

In this chapter, we present new methods to analyze the transition from menus to keyboard shortcuts. To achieve this, we first discuss theoretical and empirical characterizations of the transition from menus to keyboard shortcuts. In particular, we identify a set of behavioral markers [1] useful to precisely describe this transition and propose a methodology to numerically estimate them. We use this methodology to reanalyze the study of Grossman et al. [77].

6.1 CURRENT CHARACTERIZATION OF TRANSITION

6.1.1 Theoretical Characterization

Current theories characterize the transition from the perspectives of the definition of transition, the theoretical frameworks and the mathematical models. We are going to see how these three perspectives offer different types of characterization of the transition.

6.1.1.1 Characterization from the definition

Chapter 2 offered the following definition for transition : "The process or a period of changing from one state or condition to another.". There

1 Behavioral markers [90] aim to describe a specific observable user behavior, not an attitude or a personality trait that is linked with the users' performance

are three interesting keywords in this definition: 1. *state*, 2. *period* and *process*.

Chapter 2 discussed more in depth what can *state* be within the context of this thesis (chapter 2, section 2.2). For this chapter by *state* we mean the use of one command selection method (intermodal transition).

Period adds a temporal characterization to the transition while *process* indicates that there are some specific steps that users had to take to transition to the new method.

The definition offers a high level characterization. This characterization may not be useful to analyze users' behavior or evaluate interaction techniques since we need concrete behavioral markers to achieve these goals. It can however offer a guideline of what kind of markers to look for.

6.1.1.2 *Characterization from the theoretical frameworks*

Theoretical frameworks (chapter 2, section 2.1.0.1) like the one of expertise [39, 143] or DPL [72] offer several behavioral markers allowing to characterize the transition between two methods:

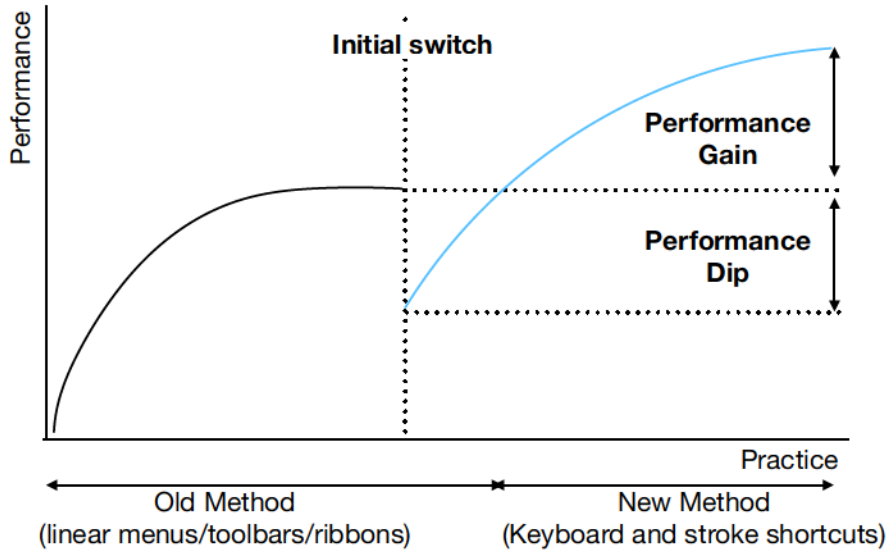


Figure 6.1: The three behavioral markers derived from Cockburn et al. [39] and Gray et al. [72]

- **Initial switch**. This marker indicates the date on which the user changes the modality.
- **Performance dip**. This marker estimates the drop in performance when changing the method.
- **Performance Gain**. The performance gain can be estimated as the ultimate performance difference between the first and the second method.

These frameworks provide a useful approximation for characterizing the transition between menus and Keyboard shortcut (KS). However, they remain theoretical approximations that must be considered with caution (e.g. does the user initiate the transition once the performance plateau is reached?).

More importantly, these frameworks suggest that the transition can be defined as a date separating the use of the first method (before) from the second method (after). This approach is fully valid when users are *forced* to change methods. For example, when they replace their QWERTY keyboard for a new AZERTY keyboard. However, in the case of commands [2], linear menus and KS coexist and users can easily switch between these two methods [96]. In particular, after an error with KS, users can reuse the menus for a while to avoid making a mistake again.

In section 6.2 we will focus more in how to use these behavioral markers to characterize transition. Part of our work is to refine the above markers to achieve a better characterization.

6.1.1.3 Characterization from the mathematical models

Mathematical models of transition to new methods (chapter 2,section 2.1.0.2 like the logistic (eq. (2.5)), Michaelis Menten(eq. (2.7)) and Saturating Exponential (eq. (2.6)) model characterize the transition by characterizing the behavioral markers that are derived from the resulted curve:

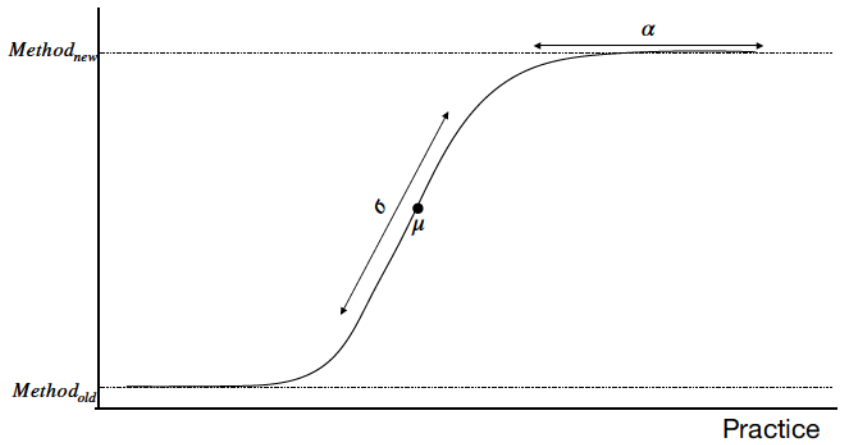


Figure 6.2: An example of the three behavioral markers that are derived from the transition models

- **Plateau of the curve (α)** : asymptotic proportion of expert method usage

2 it is also possible to have two types of keyboards at one's disposal.

- **Inflection point (μ)**: the trial for which the curve reached the 50% of the asymptotic proportion of expert method usage.
- **Scale of the curve (σ)**: which represents whether the transition is abrupt or a gradual. For an example a value close to 0 will produce a curve similar to the Constant-constant model (eq. (2.4)) which we would characterize it as an abrupt transition.

These mathematical models provide a different characterization. They measure the performance as the % of expert use (in our case % *correct keyboard shortcuts*) so they do not focus on markers like **performance dip/gain**. Also they do not provide a specific date of when the transition happened. Instead they consider that transition is a continuous process that starts from the very first time users use a system (0% of *correct keyboard shortcuts*) and continuous until they reached a plateau of performance [3].

6.1.1.4 Discussion

To summarize in this section we discussed the theoretical characterizations in current literature. There are already some differences between the characterization from theoretical frameworks and mathematical models.

The first difference is how they approach the transition period. Theoretical frameworks view this period as a specific switch date[4]. Mathematical models view this period as a continuous i.e. there is not a specific start and end date.

The second difference is the way they characterize the process. The theoretical models characterize the process based on the performance (in terms of time) drop and gain and the point of the initial change. The mathematical models characterize the process based on the curve that the models produce.

6.1.2 Empirical characterization

Numerous studies have compared interaction techniques that promote KS [77, 107, 108] or Stroke Shortcut (SS) [7, 79, 97]. The data analysis methodology that these studies follow usually include multiple steps from *cleaning the data* to *visualizing the results*.

By reviewing these studies we have detected variations in how they deal with each step of the analysis. In this section I will discuss these variations specifically focusing on how these studies deal with the *data aggregation*, *metrics* they use and *data cleaning*. These steps, especially

3 measured in % *correct keyboard shortcuts*

4 In section 6.2 we are going to introduce new behavioral markers to characterize the transition as a period.

the *metrics* can affect how researchers capture and quantify the transition.

6.1.2.1 Data Aggregation

PER USER

It is common in HCI to aggregate participants to reduce inter-participant variability. However, this practice is not recommended when studying phenomena related to learning [47, 72]. Indeed, aggregating the data of several users makes it possible to extract a "smooth" learning curve, easier to interpret, but which masks several informative behaviors such as sequences of trays, falls and performance rebounds [47, 72].

PER CONDITIONS: NUMBER AND FREQUENCY OF COMMANDS

In the same way, it is common to aggregate commands before analysis as we are not interested in inter-command differences. However, most experimental protocols studying the transition to shortcuts [7, 77, 107, 108] assign different frequencies to each command, a factor that can have a significant impact on learning [103]. The way to aggregate or not these different factors can lead to different interpretations.

6.1.2.2 Metrics

PRECISION

We identify as Precision the percentage of shortcut usage [77, 107]. See for example fig. 6.5 (A) and (B).

The first measure more accurately reflects the user's intention to adopt shortcuts. However, it does not inform if he has really succeeded in using them and therefore if he has succeeded in the transition.

In a controlled evaluation that penalizes few or no errors, the participant may be encouraged to use shortcuts, even without knowing them to save time on the experiment. In this case, it seems more appropriate to use correct shortcuts (i.e. no errors or no aid like the measure we used in Experiments 1,2,3 of chapter 5), as they show that the user is committed to the task. Otherwise, considering shortcuts (successful or not) provides more information about the beginning of the transition: abandoning menus (even with errors).

TEMPORALITY

An alternative is to describe the evolution of the percentage of use (successful or not) shortcuts over time (measured in trials or block of trials) [7, 107, 108]. See for example fig. 6.5 (D).

This representation is less concise, but more reflective of the dynamics of user behavior and may show subtle interactions between Technique and Time. For example, two interaction techniques may have the same overall percentage of use of shortcuts, but different dynamics (e.g. slow evolution vs. rapid but late progression).

SELECTION TIME

Another measure used in these studies is the evolution of selection time over time (block) . See for example fig. 6.5 (C).

This metric is particularly useful because it covers an important aspect of usability (speed). However, it does not capture transition-related phenomena such as the temporary fall in performance.

6.1.2.3 Data cleaning

INCLUSION THRESHOLDS

Finally another common approach is to use participants that used a *correct keyboard shortcuts* more than a predefined threshold. This practice was used in experiments 1,2,3 (chapter 5, sections 5.3 to 5.5). The reason to use the inclusion thresholds is to amplify the difference between the experiment's conditions by focusing on users who actually attempted to transition.

This practice though can offer misleading results. For example in fig. 6.3 we re -analyze the results from the three experiments by using different inclusion thresholds (x-axis for each subfigure). In this figure the $Included_{>=0\%}$ and $Included_{>=25\%}$ are highlighted since the three experiments used these inclusion thresholds.

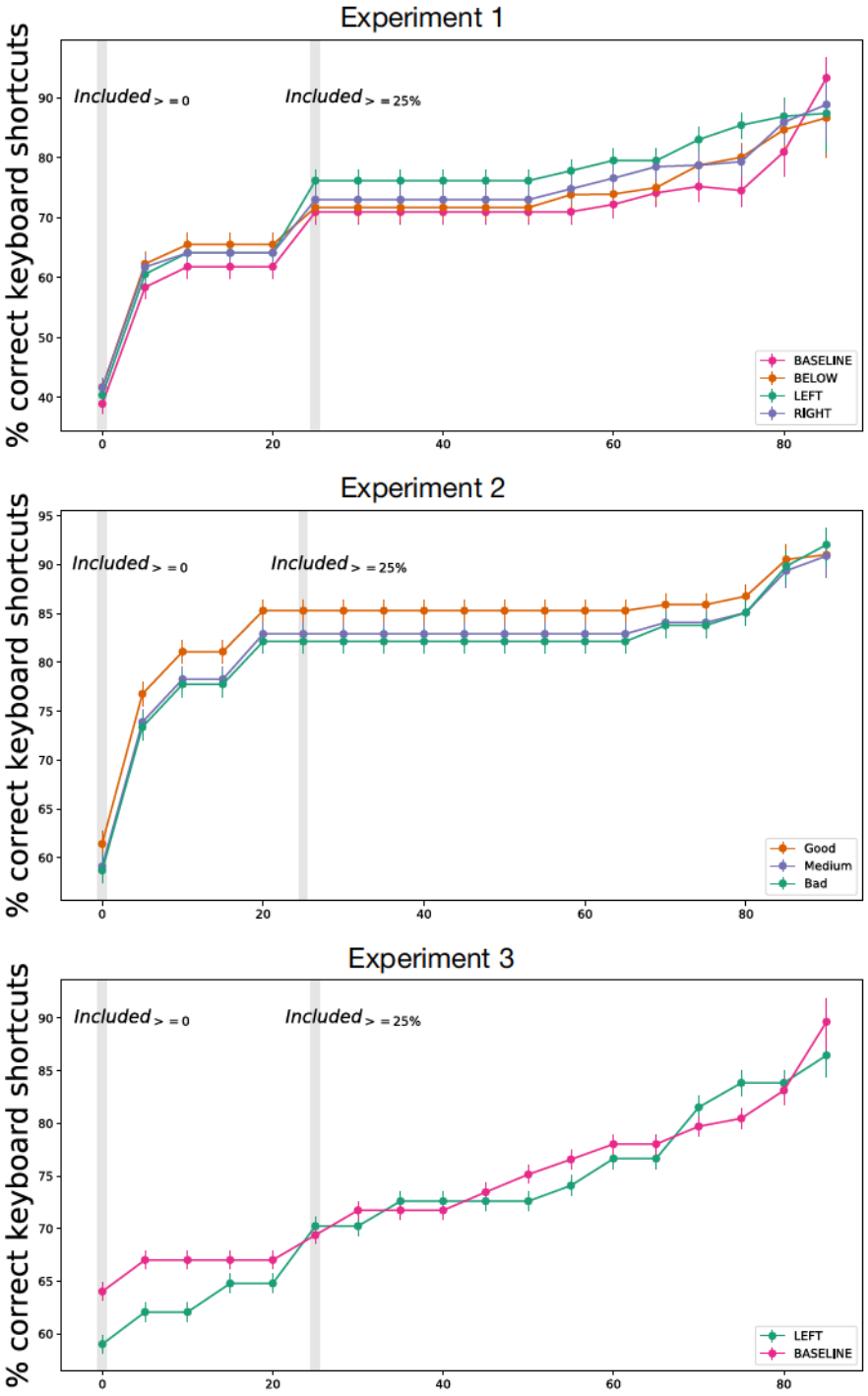


Figure 6.3

The results for this analysis highlights the problem. The inclusion threshold analysis for experiment 1 indicates that the GOOD command name- KS mapping is consistently outperforming the other conditions. In this case the choice of inclusion threshold can change the difference between the condition but not tendency of which condition tends to outperform the others. In other words all inclusion thresholds would

show that GOOD outperforms the other conditions but the "by how much it outperforms them" will vary.

In experiments 2 and 3 however the different inclusion thresholds can show different results. In experiment 2, starting from $Included_{>=25\%}$ until $Included_{>=75\%}$, LEFT layout seems to outperform the other conditions. For the rest of the cases though it is not clear if there is a difference between the conditions.

Similar behavior can be observed for experiment 3. Starting from $Included_{>=0\%}$ until $Included_{>=15\%}$, the BASELINE layout outperforms the LEFT layout. In the rest of the inclusion thresholds it is not clear if there is a difference between the two layouts.

To summarize, although inclusion thresholds can be useful to remove outliers, they can also lead to problematic practices like p-hacking[37]. So, it raises the question of how researchers should use inclusion thresholds.

To the best of our knowledge, there is not a standard way to deal with thresholds. To avoid the p-hacking problem, researchers could use preregistration or planned analysis practices [37]. These practices allow the researchers to predefine an inclusion threshold which they cannot change afterwards. These practices can lead to more reliable results. To be more transparent, researchers can present the results of an analysis next to the analysis with the inclusion threshold like Grossman et al. [77] did [5]. Another possible option is a figure like the fig. 6.3 which shows the results when we used different thresholds.

6.1.2.4 *Discussion*

In summary, there are several empirical characterizations to describe the user's ability to transition from menus to shortcuts (keyboards or stroke). Today, the main characterization is to report the evolution of the use (successful or unsuccessful) shortcuts as a function of time (or block). Although this characterization is useful, it masks several transition phenomena that can only be observed at the level of the individual such as the duration of the transition or rebound.

6.1.3 Discussion

From this discussion on the characterization of the transition between menus and shortcuts, we draw the following lessons:

- *The gap between theoretical and empirical characterizations*. In terms of behavioral markers, the former focuses on the date of change of method, the drop in performance and the gain in performance. The latter focuses on the evolution of the use of shortcuts. In terms of defining the transition, the former suggests that the transition can be perceived as a point in time, while the latter suggests

5 We followed the same strategy in experiments 1 and 2 sections 5.3 and 5.5

that the user has a continuous increase in the rate of use of shortcuts.

- *The need to analyze the data before aggregating them*. Aggregating user group/command data removes noise related to user variability, but masks learning phenomena such as tray sequences, falls and performance rebounds [72].
- *The lack of tools to characterize the transition*. It appears that current analysis strategies do not capture the menu-to-KS. In this chapter we aim to develop tools that are based on current theoretical frameworks (chapter 6).

For the rest of this chapter we will show how we can use the behavioral markers derived from the theoretical frameworks to empirically characterize the transition. I will discuss how we can use the mathematical models in chapter 7.

6.2 EXPANDING CURRENT THEORETICAL MARKERS

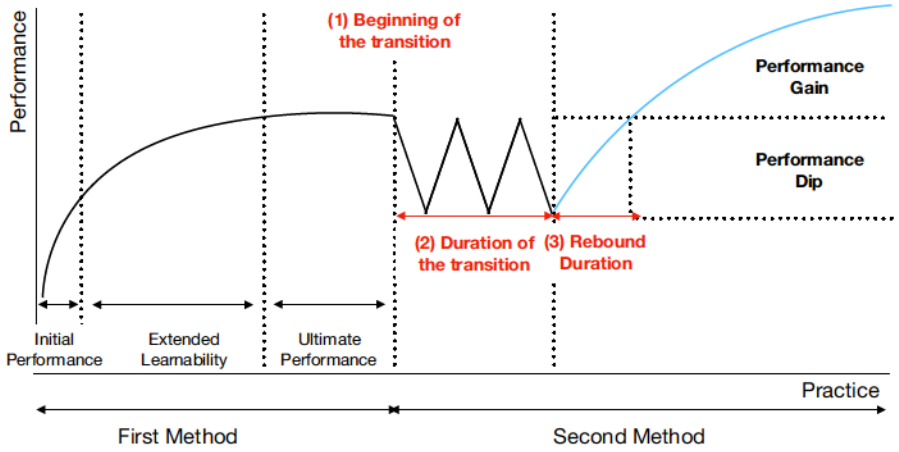


Figure 6.4: An updated version of fig. 6.1. We expanded the current markers by changing the *initial switch* marker to beginning and duration of the transition and by adding the rebound duration

The first step to use these markers is to expand them so that they can better capture the transition. Current theoretical frameworks [39, 72] consider the transition period as a single date. Yet transition happens over a period of time.

Therefore we propose a change to current characterization. In particular we propose to replace the **initial switch** marker with two new ones that can better describe the transition period.

The first one is the **beginning of transition** (fig. 6.4 - (1)). We define this marker as the date of the end of the stability period [6] with the first method which in this case is menus.

The second marker is the **duration of transition** (fig. 6.4 - (2)). It is the duration of the period of interleaving between the two methods. It ends when the users stability period starts with the novel method (in our case Keyboard shortcut (KS)).

Finally we want to introduce a new marker that is going to be useful in section 6.5. This is the **rebound duration** (fig. 6.4 - (3)). This marker indicates the time needed to regain its pre-transition performance

These three markers and the performance gain and dip can help to characterize the transition period. In sections 6.4 and 6.5 we are going to show how they can be computed and used in analyzing the users' behavior.

6.3 CASE STUDY: GROSSMAN ET AL. USER STUDY

We use the data from the Grossman et al. [77] study to investigate the different behavior markers. We first summarize this study.

6.3.1 Study overview

6.3.1.1 Experimental plan

The task was to select a target command from one of the six (non-hierarchical) drop-down menus in the menu bar or using a keyboard shortcut as quickly and accurately as possible. The stimulus was an image displayed at the bottom of the screen representing the command to be executed. An order was an animal, a fruit, a vegetable, etc. The letter used in a keyboard shortcut did not belong to the command name to avoid familiarity effects. If the user did not select the correct command, he had to execute the target command again after a 3s penalty.

Three techniques were compared: (1) TRADITIONAL: The traditional drop-down menu with visual reinforcement: the item (command name + shortcut) selected gradually disappeared while the rest of the menu was immediately hidden; (2) AUDIO: The traditional menu with audio feedback: the keyboard shortcut was played orally by a voice synthesizer once the command was executed from the menu; (3) DISABLED: The traditional menu existed but clicks in the menu were disabled to force users to use keyboard shortcuts. 42 participants (14 per technique) participated in this study.

The commands (14 in total) had different frequencies and therefore did not appear the same number of times. They could appear 144 times

6 The notion of stability is discussed in section 6.4

(2 commands), 72 times (2), 48 times (2), 36 times (4), 24 times (2) or 12 times (2) in total.

The study was divided into 12 blocks of 60 selections. The order of presentation of the commands was pseudo-random so that the frequency of each order was respected for each block (e.g. the two most frequent orders appeared 12 times per block while the two least frequent appeared once).

For each trial (or selection), *the independent variables* were the identifier of the participant, the technique, the block, the trial within the block as well as the information of the target command (label, position, frequency, keyboard shortcut). *Dependent variables* were time (ms), success (0/1), strategy used (see Terminology paragraph). All data was saved in a csv file.

6.3.1.2 *Summary of results*

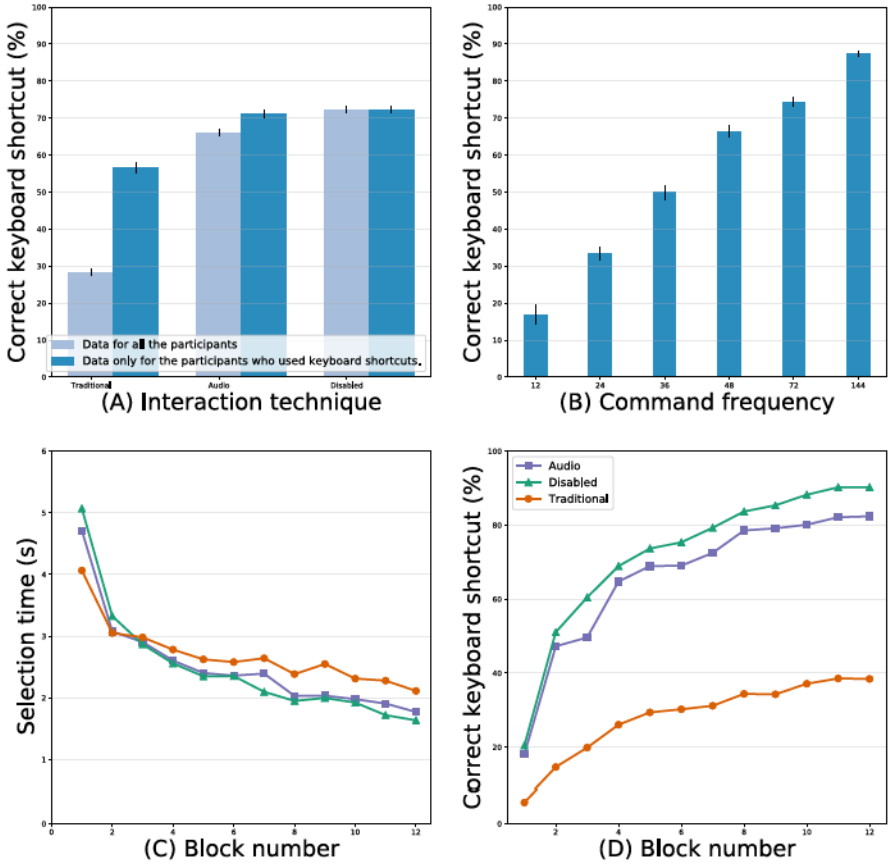


Figure 6.5: Results of the study presented in [77] . Use of keyboard shortcuts by technique and frequency with 95% confidence intervals as well as block and technique selection time.

Figure 6.5 illustrates the results of this experiment as presented in the original paper [77]. It represents in particular on the left, the percentage of use of keyboard shortcuts used (without help of the menu)

for each interaction technique differentiating the whole of the users from those which used at least once the shortcuts keyboards; in the middle, the percentage of use of keyboard shortcuts depending on the frequency of commands; on the right, the evolution of the selection time according to the technique and the block.

6.4 DATA ANNOTATION

To study the behavioral markers of the previously described dataset, we first identify the beginning and the end of the transition. As we are not aware of an automatic method on this type of data [47, 72, 161, 162], we first manually annotate the data. Afterwards the annotated data can be also used as a the base for developing and testing automatic annotation methodssection 6.4.3.

6.4.1 Problem formulation

We can better formulate the problem of identifying the transition as follows:

INPUT

As input of the annotation process, we are going to use a time series data that shows how the users behavior evolved over time. In particular we are going to use as input the user-command time series , i.e. data that represents how one user behaved while using a single command.The time therefore is measured as the number of encounters with this command.

To capture the user behavior the data includes information about how they selected the command for each trial (*strategy*) and whether they made an error.

What do we mean by strategy? Although we have two methods (menu and Keyboard shortcut (KS)) to select the command, we identified 4 different strategies (2 for each method) that users can use:

1. *menu only*: the users selected the command using the menu without pressing the keyboard.

2. *menu keyboard pressed*: the users selected the command using the menu but pressed the keyboard before.

3. *KS menu opened*: the users selected the command using the KS but opened the menu before

4. *KS only*: the users selected the command using the KS without opening the menu before.

OUTPUT

The output of the annotation process is the time series we used as input with the extra information of when (measured in number of encounters the transition started and when the transition ended.

6.4.2 Manual annotation

First we propose a manual annotation protocol to identify the beginning and the end of the transition for each time series. With this protocol researchers can go through each time series and identify the transition period. We present this process in the form of an experimental design.

6.4.2.1 Experimental plan

PARTICIPANTS

Two of the authors annotated the 588 sequences (3 techniques × 14 participants × 14 commands) first independently then collectively to resolve the conflicts.

TASK

The task was, to identify the beginning and the end of the transition for each command if **present**. According to the chapter 2, section 2.1.0.1 :

- the beginning of the transition is defined as the end of the stability period with the menu.
- the end of the transition is defined as the beginning of the stability period with keyboard shortcuts.

At this stage, there was no precise definition of the term "stability" in this context. It was left to the discretion of the annotators. Annotators, however, agree that it must be understood in terms of choice of strategy and precision.

To facilitate a consistent annotation between the annotators, they agree that the beginning of the transition can not be a successful selection using the *menu only* strategy and that the end of the transition must be a successful (i.e. no error) selection using *KS only* strategy. The impact of choice and understanding of instructions by annotators is discussed at the end of this section.

ANNOTATION INTERFACE

The participants used web application to annotate the data fig. 6.6. The application is developed in python (server) and html/javascript (client).

The application consists of multiple pages and each page shows the time series for one. Each time series is represented by dots (between

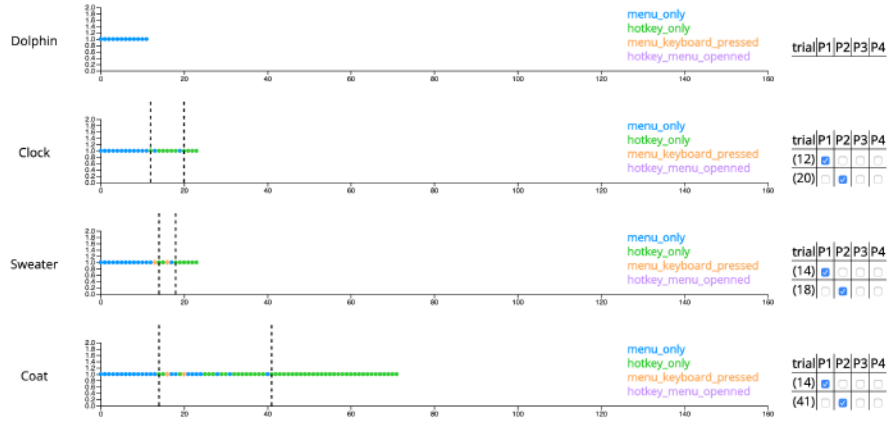


Figure 6.6: Interface for annotating a user's order. A selection is represented by point. The color indicates strategy and the symbol indicates success. Vertical lines indicate the beginning and end of the transition

12 and 144 depending on the frequency of appearance). The color of the points encodes the strategy: Blue (Menu only), Green (keyboard shortcuts only), Orange (Menu with a modifier pressed), Violet (Keyboard shortcut that opened the menu). The cross indicates an incorrect selection.

The annotator clicks on a point (i.e. a selection) to indicate the beginning or end of the transition. He can cancel or change his annotation by clicking on the corresponding point again. Vertical lines represent the transitions that are summarized in the table on the right. This is editable and allows to easily differentiate an instant transition (start == end) from the case where the user starts the transition, but without finishing it. The table also allows the users to indicate the case where the user has started a transition, but suddenly gives up and returns permanently to the menus.

6.4.2.2 *Annotators Agreement*

After independently annotating each time series, we studied the reliability of the annotations [110]

BEGINNING OF TRANSITION

the annotators had the same results for 543 of the 588 annotated sequences (92.4%). The corresponding Cohen-Kappa coefficient is 0.90 [7] The consensus between the two annotators is therefore quite satisfactory.

7 Landis et al. [19, 98] consider that values above 0.81 reflect an almost perfect consensus and Altman et al. [3, 59] consider that values above 0.80 reflect a very good consensus.

END OF TRANSITION

The annotators had the same results for 534 of the 588 annotated sequences (90.8%). The Cohen-Kappa coefficient is equal to 0.88. Although slightly worse than the one for the beginning of the transition, it remains quite satisfactory.

6.4.2.3 Resolution of disagreements

In order to use the annotations to analyze the data, the two annotators should resolve their conflicts. This means that they should find a common ground solution that satisfy both of them for all the time series they disagreed.

Therefore the two annotators discussed each time series where a conflict occurred to understand each others reasoning and try to find a solution that they can agree on. We have identified two types of resolutions: Immediate and after discussion.

IMMEDIATE RESOLUTION

One of the annotators immediately feels that the annotation of the other annotator is more relevant. This type of resolution represents 10/45 cases for the beginning of the transition and 10/54 cases for the end of the transition.

RESOLUTION AFTER DISCUSSION

Each annotator explains his choice and tries to convince the other. There were many sources of conflict. For example, should a keyboard shortcut surrounded by several menu selections be considered as the beginning of a transition or just as an isolated test? There could also be edge effects, especially for short sequences : how to consider the last selections of a sequence? To resolve these types of conflicts, annotators looked more closely at the other sequences of the same user in order to better understand their behavior. This was sufficient to agree on a decision.

6.4.2.4 Validity of annotations

To check whether the resulted annotations made sense we analyzed the degree of stability before, during and after the transition. We measured the stability as the number of strategy changes between each phase (before, during and after transition) and the percentage of errors. We expect that before and after the transition the number of strategy changes should be low as well as the percentage of errors. During the transition we expect the opposite i.e. high number of strategy change and high % of errors.

We used 95% CI to perform our analysis. Changes in strategy represent 31.5% (CI[-1.2%,+2.0%]) of encounters during the transition phase versus 2.3% (CI[-0.6%,+0.8%]) before the transition and 1% (CI[-0.3%,+0.4%])

after the transition. Similarly, the percentage of errors is 14.1% (CI[-1.4%,+1.5%]) during the transition compared to 2.6% (CI[-0.6%,+0.8%]) before the transition and 2.2% (CI[-0.3%,+0.4%]) after the transition.

These results indicate that the annotations from the manual annotation process are in agreement with our expectations.

6.4.2.5 Discussion

Like many annotation tasks, there is a degree of subjectivity in the interpretation of instructions. An important point is that although we didn't have a proper definition for the term "stability", there is a strong consensus on the results of the annotators.

Another important point is the choice of the number of phases. We had many discussions between the authors and an expert (non-author) in order to determine the number of phases to be annotated. It is indeed possible to imagine that the transition phase could be divided into several distinct phases. Among those envisaged, we can mention a phase "dominated" by the use of menus and another "dominated" by shortcuts. It was also considered to differentiate between a phase where the user "explores" the alternative method and a phase where the user is more active in learning the shortcut. In this thesis, we have chosen to consider the transition as a simple entity leaving as future work a finer decomposition of this phase. However, in the next section we will discuss the presence of a pre-transition phase.

Furthermore the manual annotation process identified the beginning and end of the transition for each command before aggregating the data by command, user, frequency or technique. These annotations make it possible to study more closely the transition between menus and keyboard shortcuts (section 6.5).

We will also use these manually annotated data to design and evaluated an automatic method. We present the corresponding algorithm in section 6.4.3.

6.4.3 Automated annotation

We now present another approach utilizing automatic annotation algorithms to (1) avoid manual annotation which is time-consuming and potentially dependent on annotators and (2) check the consistency of the annotations proposed in section 6.4.2. The first algorithm section 6.4.3.1 is a simple rolling window algorithm while the second algorithm is a neural network based solution section j.3.3.

6.4.3.1 Sliding window

As the manual annotation, the input of the algorithm is a sequence of command selections and the output is the beginning of the transition

(end of stability period with the menu) and the end of the transition (beginning of the stability period with KS).

In an ideal case, the beginning of the transition would be the first selection, which is a non pure menu selection [8]. The end of the transition would be the first correct keyboard shortcut after the last non-correct keyboard shortcut.

The reality is a bit more messy. Users can use KS before the beginning of the transition or menus after the end of the transition. To solve this problem our algorithm relies on a sliding window.

A sliding window is a list of n elements that starts from the beginning of a time series and keeps shifting one element to the right until it reaches to end of the sequence. At each step the algorithm performs an operation over the data included in the window. In our case the operation consists of counting the four different types of selections (correct/false menus/KS) to determine whether it is the beginning or the end of the transition according to different acceptance thresholds.

ALGORITHM OVERVIEW

Figure 6.7 shows how an example of how the algorithm works while algorithm 1 presents the pseudocode of the algorithm.

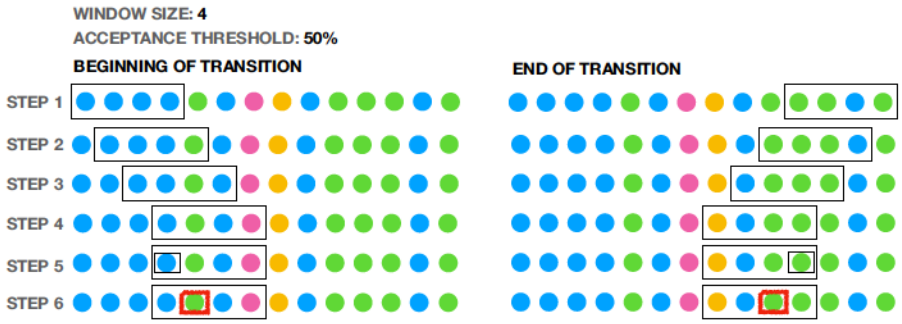


Figure 6.7: Example of how the sliding window works. In this example we used a window size of 4 sequence points and an acceptance threshold of 50%

8 by pure menu selection I mean the case where the user used menus without pressing the keyboard before regardless if they made an error while selecting the command.

Algorithm 1: Pseudocode for the sliding window algorithm

Input : Time series of encounters with a single command.
window_size
acceptance_threshold

Output : The beginning and the end of the transition

```
start window from the first element ;
initialize beginning of transition with -1;
while window is not outside of the sequence do
    sum = 0 ;
    foreach element in window do
        if element.strategy == menu_only then
            sum += 1
        end
    end
    percentage = sum*100/window_size ;
    if percentage > acceptancethreshold then
        shift window on element to the right;
    else
        foreach element in window do
            if element.strategy ! = menu_only then
                beginning of transition = element.position
            end
        end
    end
end

start window from the last element ;
initialize end of transition with -1;
while if window is not outside of the sequence do
    sum = 0 ;
    foreach element in window do
        if (element.strategy == hotkey_only) &&
        (element.number_of_errors == 0) then
            sum += 1
        end
    end
    percentage = sum*100/window_size ;
    if percentage > acceptancethreshold then
        shift window one element to the left;
    else
        foreach element in window do
            if (element.strategy ! = hotkey_only) ||
            (element.number_of_errors != 0) then
                end of transition = element.position +1
            end
        end
    end
end
```

Now we describe the algorithm in more details. The first step is to create the sequences that the algorithm uses as input. Each sequence represents the users actions for each encounter with command. We characterize the users actions by the strategy they used to select the command and accuracy i.e. if they made an error while selecting the command. We identify 4 strategies: 1. *menu only*: the users selected the command using menu without pressing the keyboard, *menu keyboard pressed*: the users selected the command using menu but pressed the keyboard, *KS menu opened*: the users selected the command using KS but opened the menu before and *KS only*: the users selected the command using KS without opening the menu before.

Then the researcher needs to define the size of the window and the value of the acceptance threshold. This step is difficult as these value directly affect the performance of the algorithm. In line 42, I will talk about how to define these values more.

Next the algorithm looks for the beginning of the transition. The algorithm starts the sliding window from the beginning of the sequence. For each window it calculates the percentage of the encounters where the users used the *menu only* strategy regardless the accuracy. If this percentage is above the acceptance threshold then it continues to the next window. If this percentage is below the acceptance threshold then one of the encounters in this window is the beginning of the transition. This encounter is the first encounter that is non *menu only* strategy. If all the windows are above the acceptance threshold then the algorithm returns -1.

Afterwards the algorithm looks for the end of the transition. This time it starts from the end and it measures the percentage of *correct keyboard shortcuts* encounters (*KS only* and no error). If the percentage is below the acceptance threshold then one of the encounters of this window is the end of the transition. This encounter is the last *correct keyboard shortcuts* encounter. If all of the windows are below the acceptance threshold then the algorithm returns -1. Furthermore if the end of the transition is one of the last two encounters of the sequence then the algorithm returns -1. This decision is based on the decisions the two annotators took during the manual annotation task.

WINDOW SIZE AND ACCEPTANCE THRESHOLD

The main caveat of this algorithm is the values of the window size and the acceptance threshold. Choosing the wrong values can have a negative impact of the algorithm's performance. The solution we currently have for this problem is to use the annotated data we have gathered to train the algorithm by identifying the values that have the most correct predictions of the beginning and end of the transition.

6.4.3.2 Machine learning algorithms

In addition to the sliding window algorithm, we collaborated with two experts in machine learning [9] to develop two more algorithms that can utilize more sophisticated technologies.

During this process I and my supervisor focused more on the conceptual aspects of the algorithms. Namely we explained the problem, provided the annotated data and discuss the various approaches.

Our machine learning experts were mostly in charge of the more technical aspects. They implemented the algorithms and created their part of evaluation process (see section 6.4.3.3).

The two algorithms were a an encoder-decoder with a 1D convolutional layer [100] and an HMM algorithm [132]. The input sequence for both algorithms was the same as the input of the sliding window algorithm (strategy/accuracy) and they identify whether each point of the sequence belongs to one of the three phases: 1. *Before the transition*, 2. *During the transition* and 3. *After the transition*. The beginning of the transition is the first point which belongs to the *During the transition* phase and the end of the transition is the first point which belongs to the *After the transition* phase. For more details see section j.3.3

For the HMM, we used 8 symbols (4 strategies combined with error/no error) and 3 states (before/during/after transition) followed by the same refinement as for the encoder/decoder.

6.4.3.3 Evaluation of the algorithms

The most commonly used method to evaluate such algorithms is the cross-validation [92]. The idea of cross-validation method is that practitioners have a dataset that includes both the data that the algorithm is going to use as well as the expected outcome. Then the practitioners use a portion of the data to test their algorithms (test-data) and the rest of their data are used to train the algorithm (training-data).

During the training process the practitioners aim to find the values for the algorithm's parameters that can optimize its performance. For example for the sliding window algorithm the training process is used to determine the optimal value of the window size and the acceptance threshold. After the training is done then practitioners can use the algorithm with the values that they found during the training phase to test it on the test-data.

This is an iterative process. In each iteration a different portion of the data is used as a test data and different portion as a training data. Usually the iterative process stops until all data have been used as test-data.

One important aspect is what kind of metric the practitioners use to evaluate the performance of their algorithms. The metrics can vary

9 Marion Morel, Catherine Achard

in each case of course, but in our case we have identify the following relevant metrics:

- *Absolute agreement* = Manual annotations == algorithm annotations. By manual annotations we mean the annotations that were produced from the manual annotation process after the conflict resolution. By algorithm annotations we mean the annotations that were produced by the algorithms.
- *Distance between annotations* = Manual annotations - Algorithm annotations
- *Agreement with at least one annotator* = (annotator 1 annotations == algorithm annotations) or (annotator 2 annotations == algorithm annotations). By annotator <x> annotations, we mean the annotations that were produced by annotator <x>.
- *Closest distance with annotators* = min(annotator 2 annotations - algorithm annotations, annotator 2 annotations - algorithm annotations),

Due to time constraints we used only the *absolute agreement* and the *Agreement with at least one annotator* metrics and we will leave the others for future work.

METHODOLOGY

We use two different types of cross-validation. The first is "one against all" (Leave one out (LOO) Cross Validation) in which we used one command as a test in each cycle of the cross validation. The second is the "one user against all" (Leave one subject out (LOSO) Cross Validation) in which we used the commands from one subject as a test in each iteration of the cross validation.

During the training phase we excluded from the train-data time-series that have less than 36 points. We took these decision because these time series are too uninformative to draw information from them (see table 6.2.

RESULTS

Table 6.1 presents the results of the evaluation for all three algorithms, for each cross validation type and each agreement type. Results show that the sliding window algorithm outperformed the other two algorithms. In the case of "absolute agreement" the success rate for identifying the beginning of transition was ≈ 97% for both the LOO and LOSO cross validation. For the end of the transition the algorithm didn't perform as well as with the beginning of the transition but still outperformed the other two algorithms. The success rate in this case was ≈ 90% for the LOO cross validation and ≈ 87% for the LOSO cross validation.

Table 6.1: Transition detection rate (in %) in cross-validation LOO and LOSO with sliding window algorithm, encoder/decoder modeling or HMM. The results are given in the form of the beginning of transition / end of transition.

Validation	Sliding Window	Encoder/Decoder	HMM
Absolute Agreement			
LOO	96.88/89.58	91.16/82.99	82.99/77.72
LOSO	96.73/87.2	92.52/82.14	82.99/77.89
Agreement with at least one annotator			
LOO	97.92/92.19	92.69/85.20	85.54/83.33
LOSO	97.62/91.67	93.71/84.86	85.71/83.33

Similarly in the case of "agreement with at least one annotator" the sliding window algorithm outperformed the other two algorithms. In this case though the success rate for all algorithms and cross validation types was higher than the "absolute agreement" case. For the sliding window, the success rate for identifying the beginning of transition was $\approx$ 98% for both LOO and LOSO cross validation types. The success rate for identifying the end of the transition was $\approx$ 92% for both cross validation types.

Finally the HMM algorithm was the one with the lower success rate scores. The success rate difference between the HMM and the sliding window for identifying the beginning of transition was $\approx$ 13% (including both cases of agreement and both types of cross validation) and for identifying the end of transition was $\approx$ 10%

6.4.3.4 *Discussion*

The above results indicate that the sliding window is the most promising algorithm out of the three for an automatic detection of the beginning and the end of the transition. Yet the success rate are not as high as we would like especially in the case of the end of the transition.

We assume that the failure of prediction is due to singular behaviors which are difficult to model. For example there are cases where the users start to transition and then they change their mind. In this case there is a transition period but is followed by menu selections.

One remaining challenge is to automatically identify these singular behaviors. From our experience with the manual annotations we have managed to identify some of these behaviors like the one described above or cases where the users transition early and then there have been periods after the end of the transition where they did an error while using the KS. A more systematic approach can potentially identify more behaviors like this which we can take into account on future iterations of the algorithms.

The main question is whether we can use these algorithms to analyze data. In our opinion the results of our evaluation are promising especially in the cases of sliding window. There is still room for improvement especially when it comes to understanding why the algorithms fail to annotate correctly the data. For example we plan to investigate whether the algorithms produce a false positive/negative, how the size of the time series affect the performance etc.

6.5 DETAILED ANALYSIS OF THE TRANSITION

Now that we have annotated the data, we can perform more interesting analysis which can answer questions regarding the transition such as: how the interaction techniques and the frequency of command appearance affected the beginning and the end of the transition? How much the performance of the users dipped and how long it took to rebound? What was the performance gain of the transition.

In this section we re analyzed the data from Grossman et al. [77] to answer these questions.

6.5.1 Data and analysis technique

For the analysis, we use the manually annotated data as they are more accurate.

Before starting the analysis we first investigated for how many commands the users started and ended the transition per frequency of command appearance. We didn't want to include sequences where the users did not start or finished the transition as it would add noise to the data.

Table 6.2 shows the number of sequences that users started and finished the transition for each technique and frequency. We observe that for commands with a frequency equal to 12 there are not enough sequences where the users begin (and therefore ends) the transition. We also observe that commands with frequency 24 there are not a lot of sequences that the users end the transition especially in the case of Traditional technique.

As a result for behavioral markers that require the beginning of the transition (all of them) we considered commands with frequency greater than or equal to 24. In addition for behavioral markers that require the end of the transition (e.g. transition duration, performance gain etc) we considered commands with frequency greater than or equal to 36.

Table 6.2: Number of commands where the user started and completed the transition by technique and frequency. In our analysis we considered the commands with frequencies that are annotated with black color.

Freq.	Traditional		Audio		Disabled		Total
	Beginning	End	Debut	End	Beginning	End	
12	4 (14.3%)	0 (0%)	14 (50%)	3 (10.7%)	28 (100%)	8 (28.6%)	28
24	21 (37.5%)	10 (17.86%)	38 (67.9%)	24 (42.9%)	28 (100%)	33 (58.9%)	56
36	11 (39.3%)	11 (39.3%)	24 (85.7%)	18 (64.3%)	28 (100%)	19 (67.7%)	28
48	12 (42.9 %)	9 (32.1%)	26 (92.9%)	24 (85.7%)	28 (100%)	25 (89.3%)	28
72	12 (42.9 %)	12 (42.9 %)	26 (92.9%)	24 (85.7%)	28 (100%)	27 (96.5%)	28
144	12 (42.9 %)	12 (42.9 %)	26 (92.9%)	26 (92.9%)	28 (100%)	28 (100%)	28
Total	68	44	140	92	140	99	

Finally to analyze our data we used 95% bootstrapped **CI!** (CI!) [50]. Especially for metrics related to time we used perform a logarithmic transformation before the analysis (and the reverse transformation after) to reduce the influence of extreme values and the asymmetry of distributions [50].

6.5.2 Results

6.5.2.1 Behavioral markers

BEGINNING OF TRANSITION

Figure 6.8 compares the different techniques for the "beginning of transition" marker. We observe an important effect of the technique on the beginning of the transition: users start the transition earlier with Audio (on average after 6 selections) than Traditional (12 selections). In the case of Disabled, the technique is designed so that the user is forced to start the transition as soon as the first selection is made. Figure fig. 6.9 suggest that we couldn't detect any effect that the command appearance frequency had on the beginning of the transition (large error bars and intersections with axis 0).

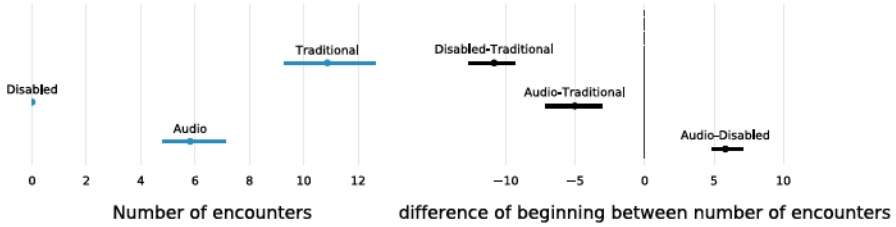


Figure 6.8: (A) Beginning of the transition measured in number of encounter with the command per technique before the beginning of the transition. (B) Differences of the beginning of the transition between the technique.

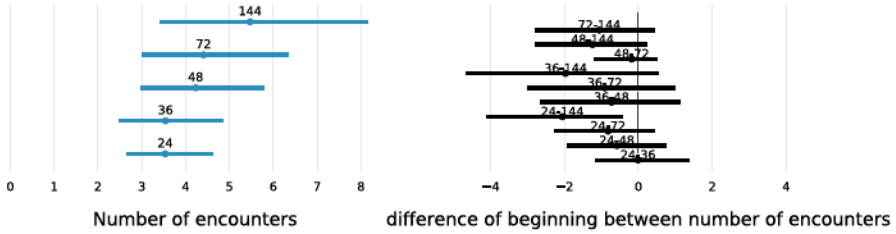


Figure 6.9: (A) Beginning of the transition measured in number of encounter with the command per command appearance frequency before the beginning of the transition. (B) Differences of the beginning of the transition between the command appearance frequencies

DURATION OF TRANSITION

Figure 6.10 compares the duration of the transition by technique. It suggests that we couldn't detect any effect that the interaction techniques had on the transition duration. Similarly fig. 6.11 indicates that we couldn't detect any effect that the command appearance frequency had on the transition duration.

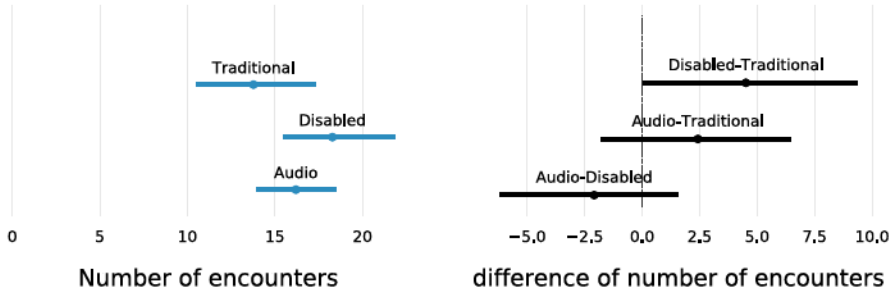


Figure 6.10: (A) Duration of the transition measured in number of encounter with the command per command appearance frequency. (B) Differences of the duration of the transition between the command appearance frequencies.

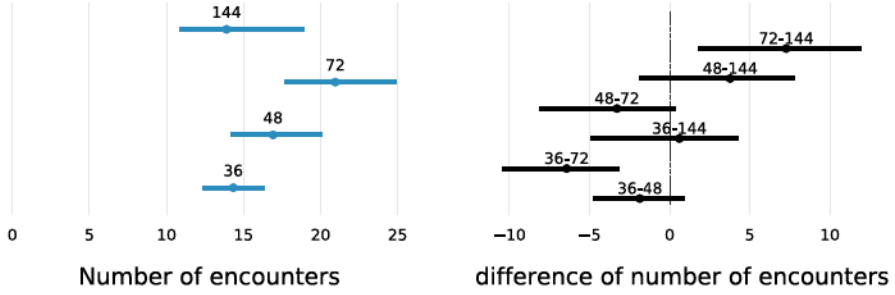


Figure 6.11: A) Duration of the transition measured in number of encounter with the command per command appearance frequency. (B) Differences of the duration of the transition between the command appearance frequencies.

PERFORMANCE GAIN

To estimate the performance gain, we propose to measure the difference in the best selection time before the beginning and after the end of transition phase for each sequence. Because of the variability from one selection to another, we observed that it was more relevant to consider the two fastest *consecutive* encounters and to take the average. Figure 6.12 shows that the performance gain is 1.3s (CI[+/- 0.3s]) on average and does not depend on technique or frequency.

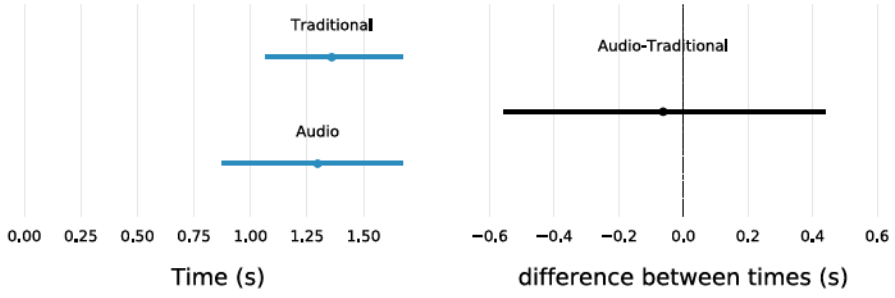


Figure 6.12: (A) Performance gain for each technique measured in seconds. (B) Difference of performance gain for each technique.

PERFORMANCE DIP

While the selection time with menus should decrease with practice [117],fig. 6.16 suggests a gradual increase in the selection time as the transition approaches the beginning.

We then propose to measure the difference between the selection time at the beginning of the transition (average time of the first two encounters) and the best time before the transition (average of the two best consecutive times). Figure 6.13-A-B indicates a 1.4 second drop in performance (CI[-0.5,+0.6]) that does not seem to depend on the interaction technique (Inactive is not considered here because it does not have a Pre-Transition phase).

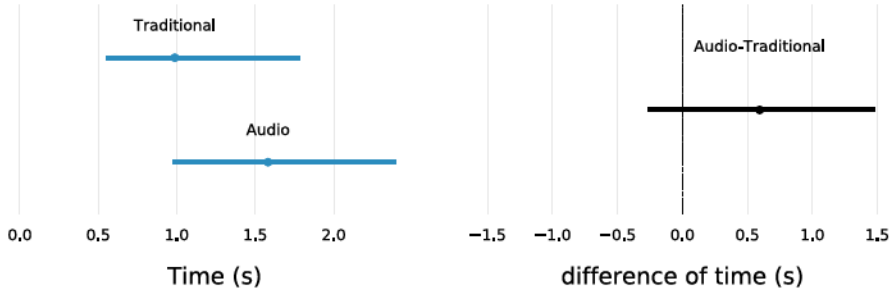


Figure 6.13: (A) Performance drop for each technique measured in seconds. (B) Difference of performance drop for each technique.

We refine the study of the drop in performance by analyzing the few encounters preceding the beginning of the transition. More precisely, we analyze the period between the best selection with the menu and the first selection of the transition phase that we call the *pre-transition phase* (fig. 6.14). The duration of this period is on average 2.9 (CI[-0.4,+0.5]) encounters with Audio and 5.4 (CI[0.9,1.2]) with Traditional.

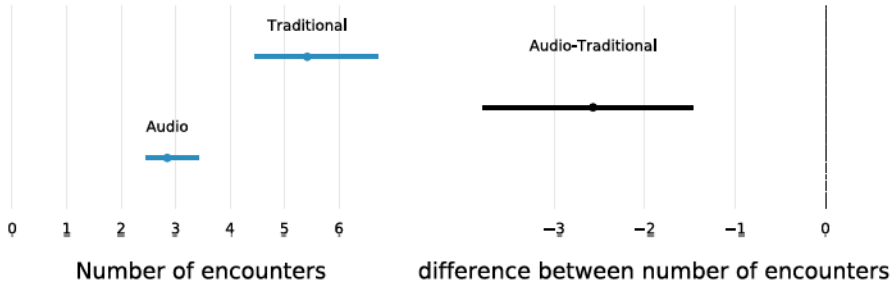


Figure 6.14: **(A)**: Number of command encounters between the best time before the beginning of the transition and the performance when the transition started. (B) Mean differences.

REBOUND DURATION

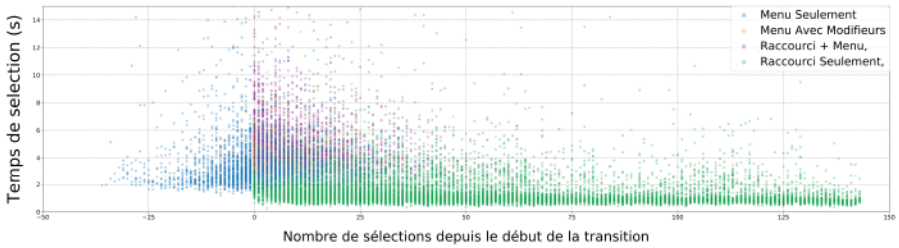


Figure 6.16: Representation of all command sequences where the user has started a transition. The sequences are aligned with the transition date. Color encodes strategy

We defined the duration of the rebound as the time required to return to pre-transition performance. We measure it as the number of encounters separating the best selection before the transition and the first faster selection after the transition [10] (fig. 6.15). The rebound duration is 18 encounters (CI[+/- 2]). A visual analysis of each sequence suggests that the end of the rebound period often coincides with the end of the transition. Indeed, the number of encounters between the end of the transition period and the end of the rebound period is equal to 0.6 encounters (CI[-0.3,+0.6]) for Traditional and 0.3 encounters (CI[+/- 0.1]) for Audio.

6.5.2.2 *Additional Analysis*

In addition to the analysis for each behavioral marker we also perform the following analysis

SELECTION TIME

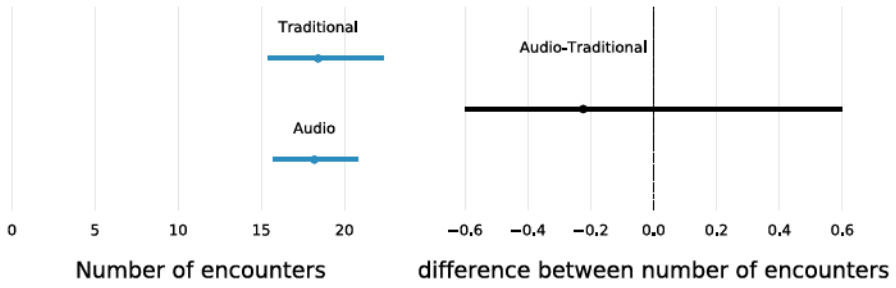


Figure 6.15: (A) Rebound duration per technique. (B) Mean difference of rebound duration between the techniques.

Figure 6.16 represents the selection time according to the number of encounters since the beginning of the transition. The color encodes the strategy used. We detail this figure through 2 behavioral markers.

10 In both cases, we consider the average time of two consecutive encounters

Figure 6.17 represents the distribution of the selection time (A) before and after the transition and (B) during the transition. For the selection time with the menus in Figure A, we distinguish between users who have started the transition and those who have not started the transition.

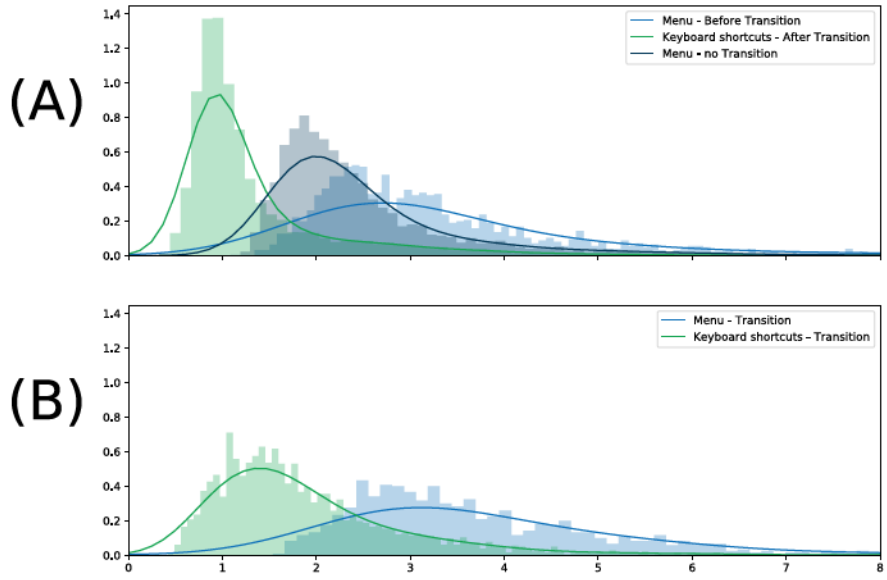


Figure 6.17: Time distribution before the beginning and after the end of the transition for menus and Keyboard shortcut (KS) (A) and during the transition(B).

There is little overlap between keyboard shortcuts on the one hand ($median = 1.2; CI[+/-0.2]$) and menus ($median = 3.7s, CI[-0.4, +0.2]$) on the other hand, even for users who have only used menus ($median = 2.4, CI[+/-0.2]$) (fig. 6.17 (A)). This overlap is more important during the transition (fig. 6.17 (B)), mainly because the distribution of keyboard shortcuts extends more to the right.

We also observe that users who have not made a transition are faster with menus than those who have started the transition, probably because they have been able to practice this method more (learning curve).

6.5.3 Discussion

Our analysis presents a new characterization of the transition from menus to keyboard shortcuts and offers a new perspective on the development of inter-method expertise. In particular, it suggests that:

1. There is clearly a transition phase for most of the selection sequences of a command. This transition is rarely instantaneous as suggested by current theories [39, 72], and does not extend throughout the experiment as suggested by the empirical characterizations.

2. Interaction techniques have an impact on the beginning of the transition (AUDIO: 6 encounters; TRADITIONAL: 12 encounters), but not on its duration (17 average encounters).

3. The variable NUMBER OF ENCOUNTERS seems to explain the beginning and duration of the transition better than the FREQUENCY. Frequency is a complex factor [103]: it modifies not only the number of encounters for a target command, but also the number of commands interspersed between two consecutive target commands. We initially thought that the user would be more motivated to start the transition if the same command appeared several times in a short period of time. For example, we thought that if the same command appeared three times in a row, the user would start a transition. This does not seem to be the case. On the contrary, the number of encounters seems to be more explanatory.

4. The PERFORMANCE GAIN observed in this study is 1.3s suggesting a benefit to using keyboard shortcuts. However, the reader may question the measure used for this behavioral marker: the time difference between the fastest selection BEFORE and AFTER the transition. There is no evidence that the user was able to achieve the performance plateau with either method. In fact, (1) it appears that menu users are faster if they never make transitions, suggesting that many users start the transition before they reach the menu performance plateau. This is interesting because it is not the lack of progress with a method that motivated users to start the transition; (2) Regarding keyboard shortcuts, it is not clear whether users managed to reached the plateau of performance.Further analysis using relevant models [118] could provide more information regarding the plateau of performance with KS and whether users managed to reach it. In summary, therefore, our measurement does not accurately reflect the behavioral marker as described in the theoretical characterizations. However, these do not seem to reflect the behavior of users; users frequently start the transition before they have reached a performance level. Our measure is therefore more appropriate for our observations, but should be renamed *performance gain before and after transition*.

5. The drop in performance has often been cited as a reason why users did not make the effort to learn keyboard shortcuts, but to our knowledge had never been estimated. Our results suggest a drop of about 1.4 seconds. This analysis also revealed the presence of a pre-transition phase.

6. the transition is preceded by a phase of PRE-TRANSITION characterized by the fact that the selection time with the menus increases.

We think that the user prepares his transition during this sentence by spending more time in the menus to read/learn keyboard shortcuts. The duration of this phase seems to depend on the interaction technique: 2.9 encounters for Audio versus 5.4 for Traditional. This suggests that the user needs to read/learn more keyboard shortcuts with Traditional because of the lack of advanced feedback. An occulometric study is necessary to better understand the intentions of users. It would not only make it possible to analyze the evolution of the locus of attention during the experiment, but also to estimate cognitive effort. We also plan to use brain imaging (EEG) and methods to encourage users to express their behavior to capture hesitations, for example.

7. Finally, we estimated the duration of the rebound at 18 encounters. It is interesting to note that the end of the transition coincides with the end of the rebound period.

Finally, although not presented here, we performed an analysis using these annotations from the sliding window algorithm. The results follow the same trend apart from the results of the transition duration where we saw some cases where the frequency of command appearance had an effect on the transition duration. This difference can be explained due to the percentage of success rate regarding the transition end.

6.6 CONCLUSION

In this chapter, we analyzed and characterized the transition from menus to keyboard shortcuts. We showed that we could identify the beginning and the end of the transition at the command level and better understand the impact of interaction techniques on users' behaviors. For instance, we saw that the interaction techniques in [77] affected the beginning but not the end of the duration. We also gained more insights about the performance dip/gain and the rebound duration. However, future work should study standardized methods to estimate these behavioral markers. While our automatic annotation is a promising approach, we need to validate it on different data sets.

Part IV

GENERAL DISCUSSION AND CONCLUSIONS

7 CONCLUSION

7.1 SUMMARY

Helping users to adopt more efficient behaviors is not an easy problem. In the context of the command selection task, it requires among others, to understand how users behave while using an interface, how their performance evolves before, during and after the transition and how the interface can help them to implicitly and explicitly reach to an optimal behavior.

In this thesis I focused on how to design interfaces that better promote Keyboard shortcut (KS) and how to characterize the transition from menu to KS. In particular, I focused in three more concrete subgoals:

- *Explore the design space of KS to identify potential designs.*
- *Designing interaction techniques that promote KS usage*
- *Implement methodologies to characterize the menu-to-KS transition*

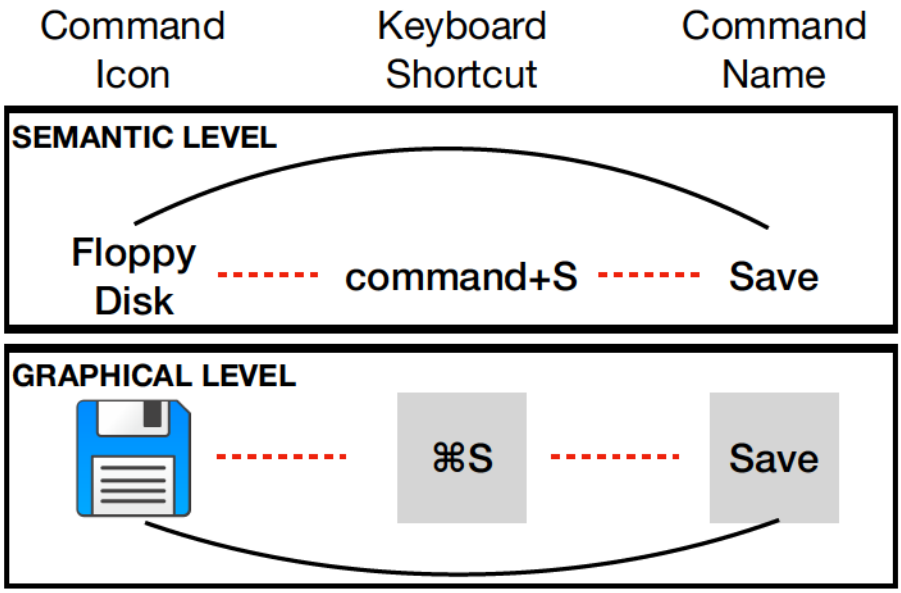


Figure 7.1: In this thesis we focused on improving the semantic and graphic relationship between the KS and the other two elements (red dotted line).

To explore the design space of KS, we first studied current practices to design the command name , the command icon and the KS focusing on

their semantic and graphic relationship (see chapter 3 for more details). We saw that when it comes to KS design there is space for improvement in both types of relationships with the other two elements (fig. 7.1). Based on current theories like Implicit Statistical Learning (ISL) (see chapter 2), we believe that by strengthening the relationships among the three elements, an interface can better promote the shortcut usage.

The findings from the exploration of the design space motivated the work we did to achieve the second goal, i.e. *design interaction techniques that promote KS*. First we proposed IconHK (chapter 4) which offered a new perspective on icon design that visually blends the command icon with the KS cue (fig. 7.2-Left). Afterwards in chapter 5 we explored how the command name-KS mapping (fig. 7.2-Right) and distance between the two elements (fig. 7.2-Middle) can favor KS usage.

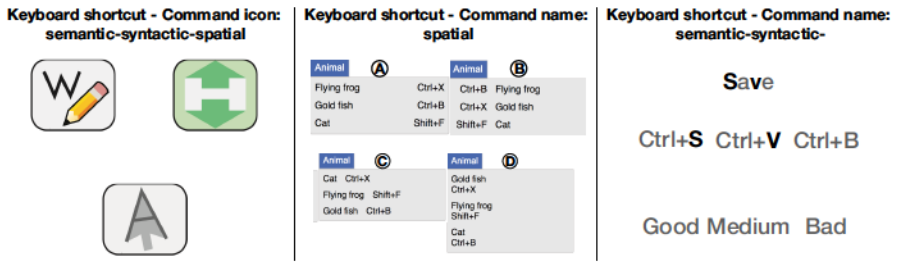


Figure 7.2: **Left**: IconHK (chapter 4 aims at improving the relationship between the command icon and the KS. **Middle**: The exploration of the position of the KS cue (chapter 5) aimed to improve the graphic relationship between the command name and the KS. **Right**: The exploration of the command name-KS mapping aimed at improving the semantic-syntactic relationship between the command name and the KS.

Results from the evaluation studies of these techniques are promising but the magnitude of the effect is not clear. This is especially true for the studies in chapter 5 which test the interaction techniques regarding the relationship between the command name and the KS in more ecologically valid settings. We still believe that we can derive interesting design guidelines from ISL theory but there is still the need to understand many things regarding this theory (see section 7.2)

One problem that we encounter during the evaluation of our studies is how to characterize and analyze the transition period (chapter 6). This problem motivated us to seek new methodologies that researchers can use to analyze empirical data and understand how users transition from one method to a more efficient one.

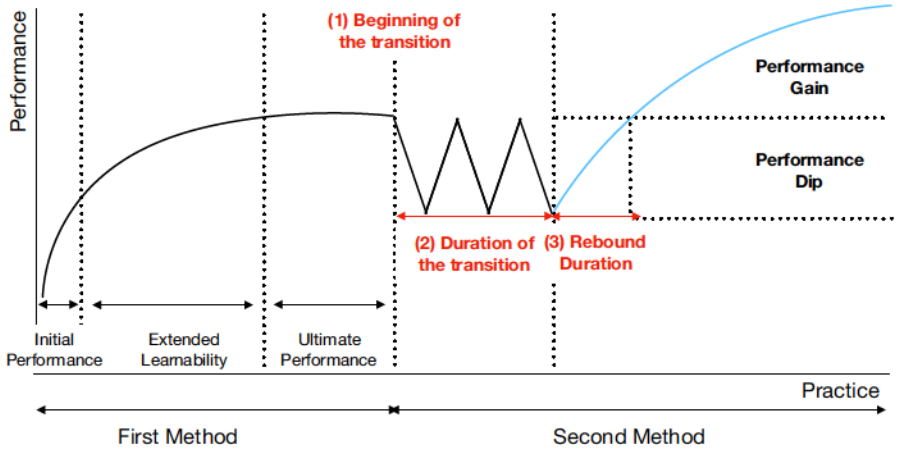


Figure 7.3: To characterize the transition (chapter 6 we used the following markers: 1. *Beginning of the transition*, 2. *Duration of the transition*, 3. *Performance dip*, 4. *Performance gain* and 5. *Rebound duration*

We based our methodology on current theories regarding transition (chapter 2). From these theories we derived behavioral markers that can help us characterize the transition (fig. 7.3). Afterwards we proposed two approaches to identify the transition period by either manually annotating the data or by using algorithms for automatic annotation. Finally we showed how by analyzing these behavioral markers we can gain a better understanding how users behave and how an interaction technique affects their behavior.

To summarize, this thesis aims to motivate researchers, practitioners and designers to question current practices of interface design and data analysis. However, there is still work to be done in order to fully realize the goals of this thesis. In this chapter I address the current limitations of the work I presented and future work directions.

7.2 FUTURE WORK AND PERSPECTIVES

I see two main paths where I would like to focus to expand the work that has been done in this thesis:

1. *Designing interfaces that support transition to more optimal methods.*
2. *Understanding and characterizing the transition to the more efficient method.*

7.2.1 Designing interfaces that support transition

EXPANDING THE DESIGN SPACE

A short term goal is to expand the current design space regarding the design of the command name, the command icon and the shortcuts. In

particular I want to systematically define the properties for each type of relationship.

Regarding the graphic relationship I aim to start by defining the spatial, temporal and appearance attributes for each element. During this thesis I have already discussed about these properties in chapter 3 and the techniques presented in chapters 4 and 5 build upon those properties. But I aim to properly define them in way that a designer can use the design space to generate better interfaces

Another property that is worth exploring is the accumulation of information (see chapter 2). Recent results in statistical learning underline the impact of *Accumulation* on incidental learning [153]. Therefore it is interesting to explore how the number of keyboard shortcut cues and icons displayed in the menu can influence the transition to keyboard shortcuts.

Afterwards I aim to expand the design space to include Stroke Shortcut (SS) which is a promising modality in Graphical User Interface (GUI). Therefore with this design space practitioners can get inspirations on how to integrate the SS properly to modern interfaces.

Finally a long term goal is to explore other types of menu systems either in the personal computers or in other devices like smartphones and tablets. Current literature has proposed different menu layouts [2, 36] as well as menus that are based on gestures [12, 96]. In both cases, by examining these menus under the dimensions of the design space, practitioners could better understand their advantages and disadvantages.

TOWARDS A DESIGN FRAMEWORK FOR IMPLICIT AND EXPLICIT LEARNING

One of the motivation for exploring these dimensions is the role of learning on the transition, especially the role of ISL. Although in our studies we see that our techniques promoted the KS usage we still miss information about how these techniques affect the transition (e.g. they affect the beginning the transition, the duration etc).

A short term plan is to annotate the data of experiments 1,2 and 3 from chapter 5. By following the same methodology as the one from chapter 6, we can better understand how ISL affects the user behavior.

Another plan is to focus on experimental design. One issue we faced on chapter 5 is how to design such an experiment that is ecologically valid and can provide information about ISL and the transition.

In the current experiments the users used the interface during a single session lasting between 45-90 minutes (depending on the experiment). It is interesting to see how the users will behave when they have to use the interface for more time i.e. multiple sessions.

Furthermore I want to identify proper participant recruitment protocols to ensure more accurate results. One possible issue of experiment

3 (chapter 5, section 5.5) was the participants' profile. A proper protocol can potentially prevent such issues in the future.

In addition, I aim to run a more ecologically valid experiment for IconHK. The current experiment differs from the experiments of chapter 5. I aim to run an experiment for IconHK similar to experiments 2 (section 5.4) and 3 (section 5.5). These experiments are closer to an actual command selection task therefore it is interesting to see how well IconHK performs under this setting.

The long term goal though is to propose proper design guidelines on how to design interfaces that support both implicit and explicit learning. To achieve this goal, apart from the previous steps I discussed, I aim to investigate more in depth the role of explicit learning by reviewing current literature and identifying potential gaps worth exploring. Ultimately this work could lead to a set of design guidelines that can inspire and guide designers when creating new interfaces.

REAL WORLD DEPLOYMENT

One important question is whether iconHK icons and menu layouts like LEFT layout can be designed easily. In chapter 4 we presented an example of the photoshop palette (fig. 7.4-A) embedded with IconHK icons. However the results from the designers study section 4.4 indicated that creating such icons can be a complex procedure. To that end we created a prototype of a tool called IconHK Maker to help designers with the design process. I aim to improve this algorithm by using modern machine learning techniques. In addition I want to work closer with designers to identify what other features may be useful to help with the design process of IconHK.

Concerning the novel layouts design, in chapter 5 it can be useful to conduct a similar study like the one for the IconHK because professional designers can better pinpoint the possible difficulties.

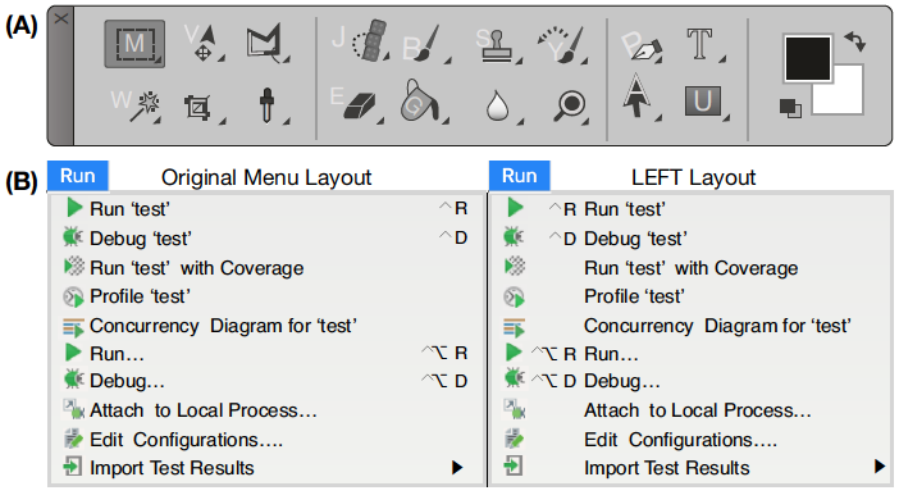


Figure 7.4: (A) Reimagining the photoshop palette using IconHK icons (for more details see chapter 4). (B) Reimagining the Pycharm linear menu by positioning the KS cue on the left of the command

One problem with designing these menu layouts is the current graphical toolkits. Common frameworks (e.g. Qt [131] , Java Swing [121] and Electron [51]) help developers implement menus on desktop applications. However, they do not provide an API to easily change the location of the different elements in a menu item. Other frameworks offer flexibility regarding the location of the elements. For instance, I implemented the context menu of experiment 3 (chapter 5, section 5.5 using the jquery contextmenu [62] library. To implement the LEFT, I used their *custom command* feature [62] to define our own menu item layout requiring 10 lines of code.

Therefore in the long term, a possible evolution of IconHK Maker is to become a tool that can help designers and developers to create novel menu systems. Such tool can hep professionals to implement toolbars and menus that incorporate IconHK as well as the novel menu layouts like LEFT, RIGHT and BELOW.

7.2.2 Understanding and characterizing the transition

CONTINUING THE ANNOTATION OF THE DATA

As I mentioned in the previous section, one direction is to annotate the data from experiments 1,2,3 from chapter 5. This process can provide better insights on how the various conditions from these experiments affect the participants' behavior.

In addition the annotated data can be used to further validate the automatic annotations algorithms. A first step is to validate the performance of the algorithms based on the results of the cross validation training. Afterwards the newly annotated data can be used to further train the algorithm.

AUTOMATIC ANNOTATION ALGORITHM

After I complete the annotation of experiments 1,2 and 3 from chapter 5 I plan to use the data to further evaluate the algorithms and see how well they perform on other data. From this process, I aim to see whether researchers can use the algorithms to analyze data and understand in which cases they fail.

To achieve that I want to use other metrics to evaluate the algorithms. In chapter 6 I presented various metrics but I used only two of them (*Absolute agreement*, *Agreement with at least one annotator*). The other two metrics (*Distance between annotations*, *Closest distance with annotators*) can also help to understand how to improve the algorithms.

One step towards this direction is to consider the behavior of the user as a whole.The current algorithms consider only one command without taking into account how the user behaved with the other commands. When resolving the conflicts of the manual annotation this tactic was useful to take a final decision and it could be useful for the automatic annotation process as well.

CHARACTERIZING THE TRANSITION BY MATHEMATICAL MODELS

During this thesis, I presented a novel methodology to characterize the transition based on current theories chapter 6. However, mathematical models can also provide interesting characterization for the transition.

The main issue is how to fit the data into the model. The most straightforward way is to fit the data for each condition separately. There is a risk of overfitting with this approach however. If there is an experiment with 4 conditions like Experiment 2 of chapter 5 which compare 4 menu layouts, it could mean that we have to fit a model of 12 parameters (4 conditions x 3 parameters).

The solution we are working on, tests different cases. In each case we fix one or more of the parameters for each regression (one regression per condition). We then compare each case by using the Akaike Information Criterion (AIC) [82] values [1] and we choose the best one.

So far we have tried this methodology using the logistic model (see chapter 2, section 2.1.0.2) on experiments 1,2 and 3 from chapter 5. Results have been mixed mainly because we are trying to see whether the outcome of this methodology is interpretable.

I plan to keep working on this methodology until it reaches a level that can be properly used by researchers.

MODELING THE TRANSITION

Another long term goal is to develop a model of performance that describe the transition.

1 The AIC value estimates the quality of a model in comparison to other models

A model of transition which is in the same spirit of the menu performance models [18, 38] can be a useful tool for practitioners. Such a model can be used to describe empirical data and predict future behaviors [85].

An ideal model synthesize complex phenomena like the users' awareness of the new method, perception of the future performance with new method etc into a mathematical equation. Current models of transition (chapter 2) characterize the curve of the transition but they do not capture such phenomena.

Part V

APPENDIX

h APPENDIX ICONHK

Figures h.1 to h.4 shows the icons that the designer who participated in our study produced.

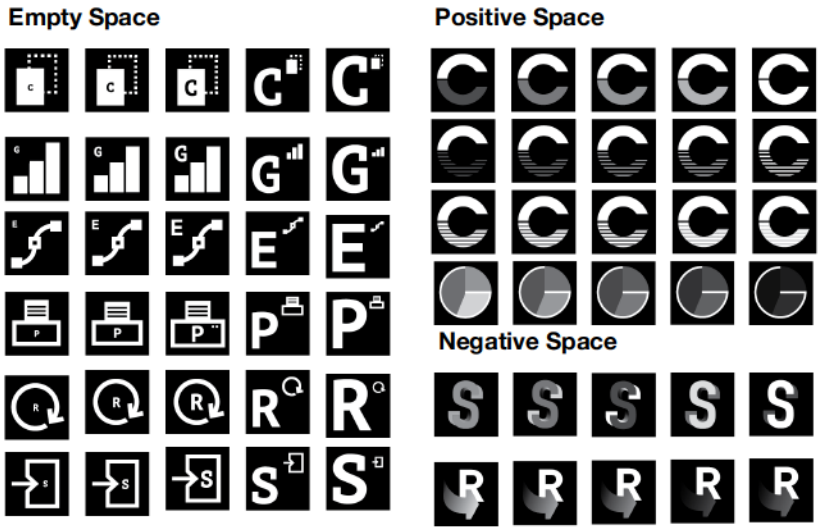


Figure h.1: Icons that D1 produced

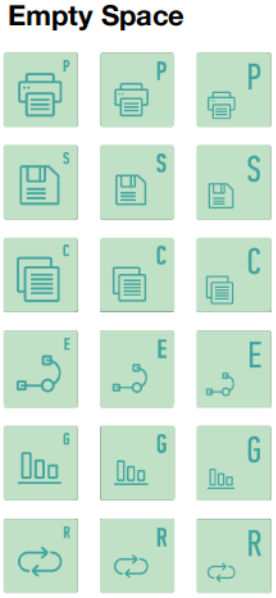


Figure h.2: Icons that D2 produced

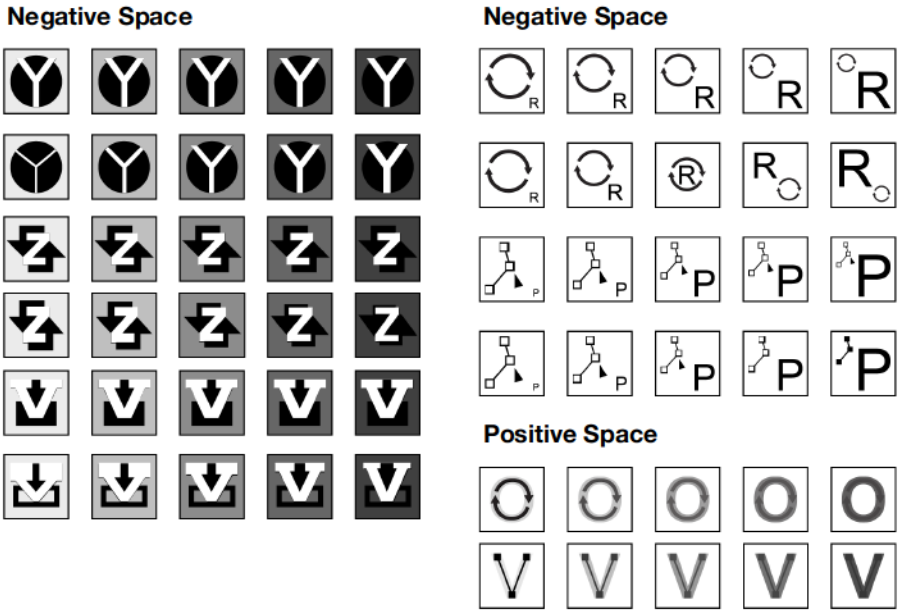


Figure h.3: Icons that D3 produced

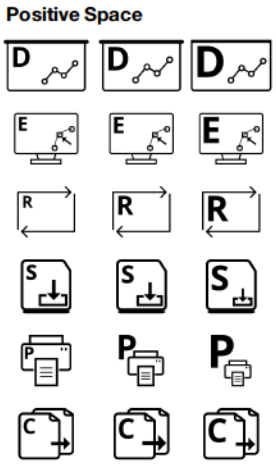


Figure h.4: Icons that D2 produced

i APPENDIX: KEYBOARD SHORTCUTS IN MENU

I.1 DESIGNER RESPONSES

Figure i.1 shows the menu layouts that designers had to evaluate. Figure i.2 shows the average rate for all the designers and fig. i.3 shows the average reate for the 4 menu layouts we evaluated in experiments 2 and 3 in chapter 5.

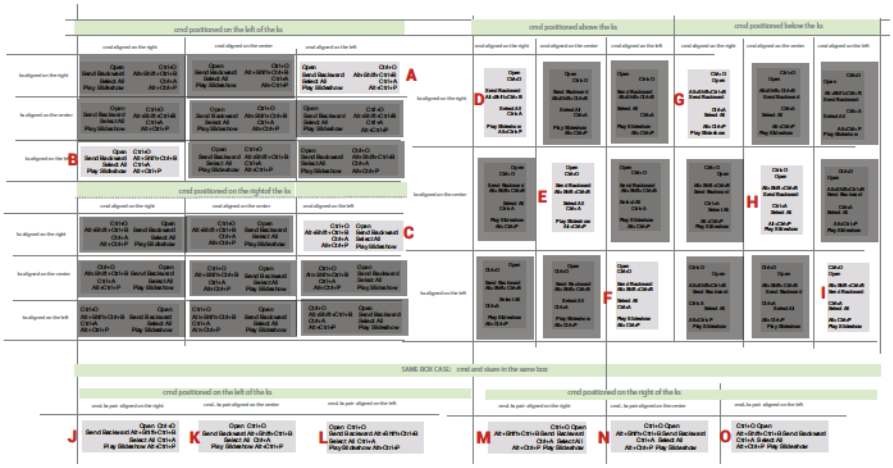


Figure i.1: The menu layouts that the designers had to evaluate.

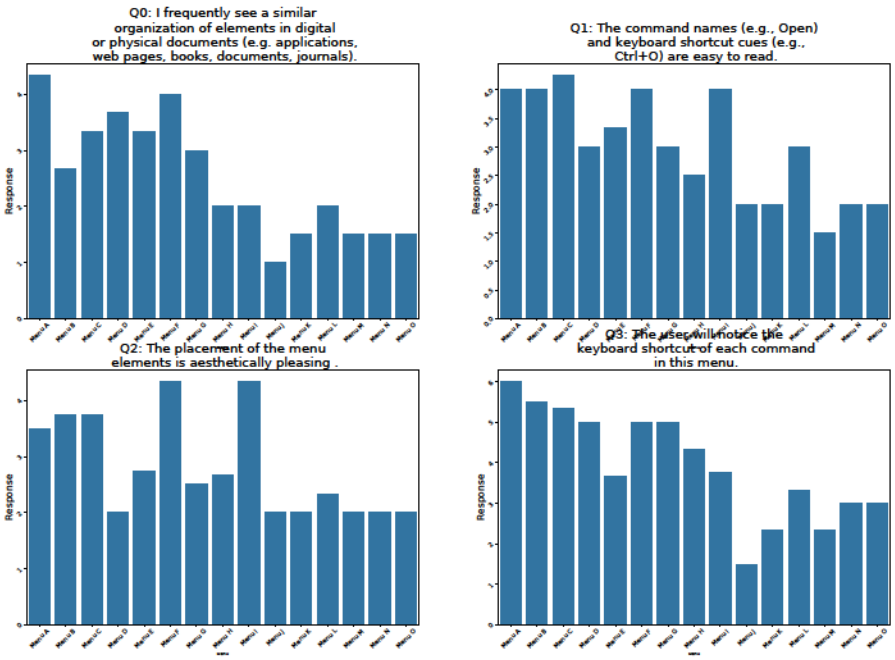


Figure i.2: Designer responses for all menu layouts

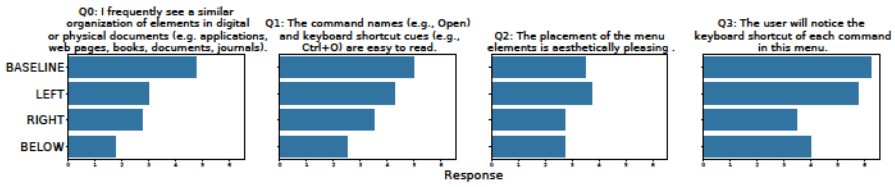


Figure i.3: Designer responses for BASELINE, LEFT, RIGHT and BELOW layouts

I.2 EXPERIMENT 1: FURTHER ANALYSIS

ERROR ANALYSIS PER MENU SYSTEM

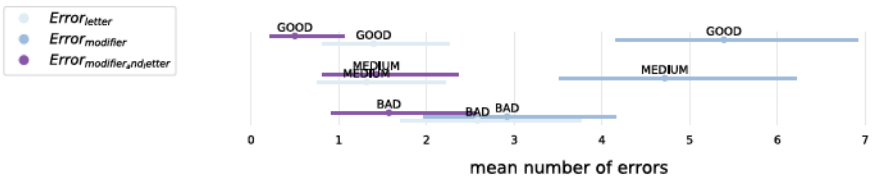


Figure I.4: Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs.

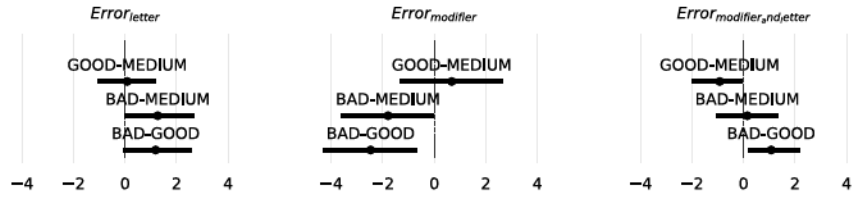


Figure I.5: Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs.

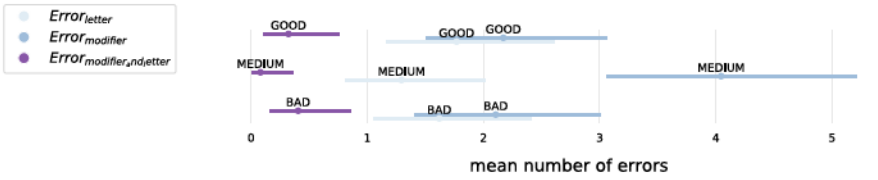


Figure I.6: Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs.

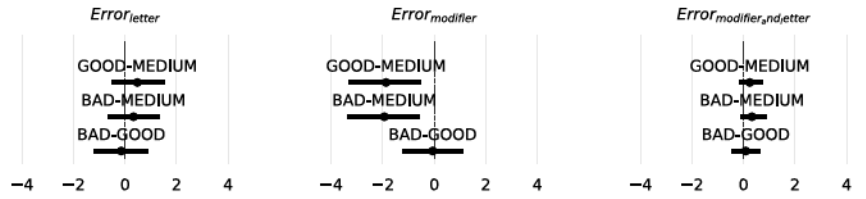


Figure I.7: Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs.

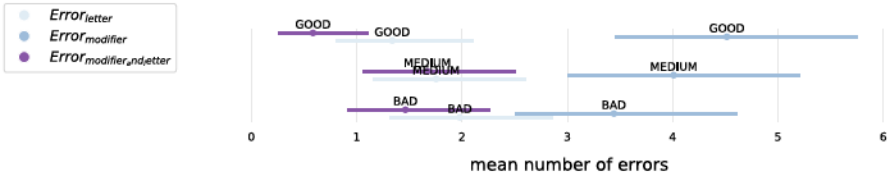


Figure i.8: Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs.

Figure i.9: Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs.

I.3 EXPERIMENT 2: FURTHER ANALYSIS

Figures i.10 to i.17 show the results of the command length analysis for each layout and each threhold level. As stated before for LEFT and BELOW the command length did not affect the shortcut usage. For RIGHT and BASELINE though the MEDIUM under-performed in comparison to the other two conditions

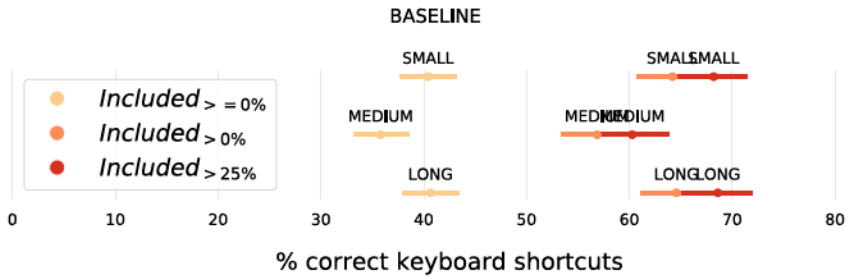


Figure I.10: Mean rate of *correct keyboard shortcuts* per layout for each threshold for BASELINE. All error bars are 95% CIs.

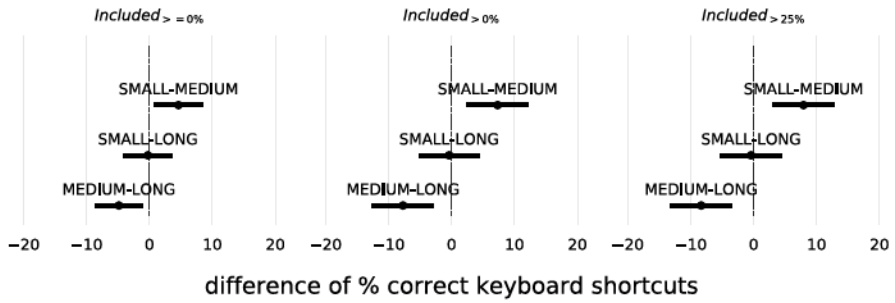


Figure I.11: Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for BASELINE. All error bars are 95% CIs.

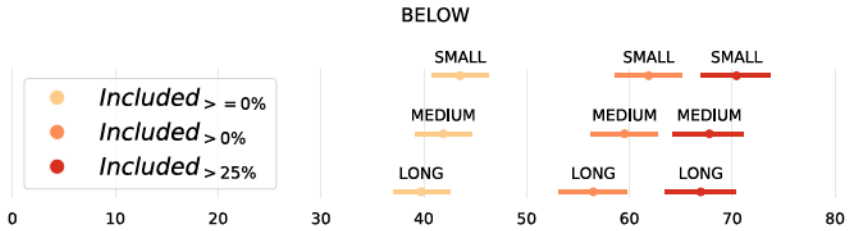


Figure I.12: Mean rate of *correct keyboard shortcuts* per layout for each threshold for BELOW. All error bars are 95% CIs.

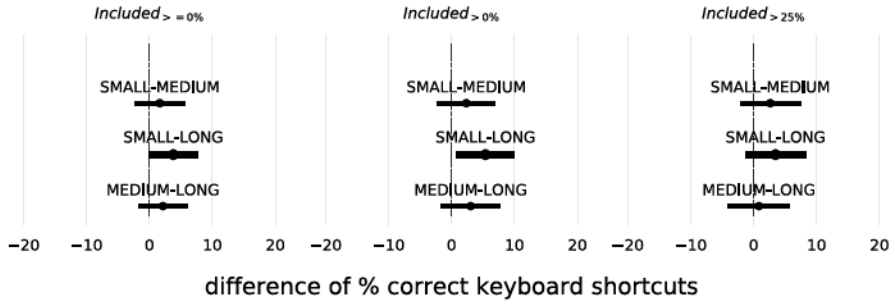


Figure I.13: Mean rate difference of *correct keyboard shortcuts* between between menu item lengths for each threshold for BELOW. All error bars are 95% CIs.

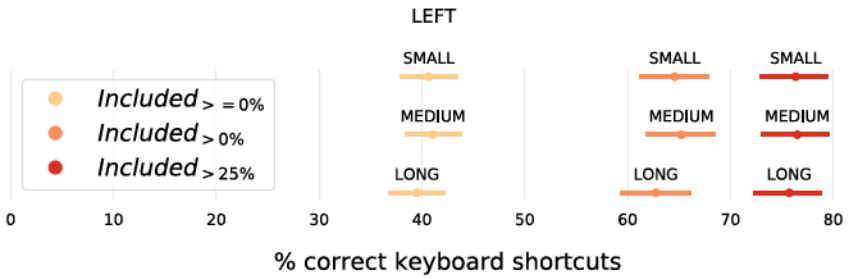


Figure i.14: Mean rate of *correct keyboard shortcuts* per layout for each threshold for LEFT. All error bars are 95% CIs.

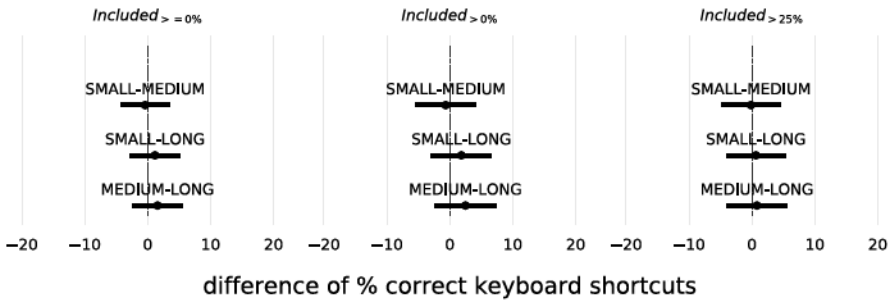


Figure i.15: Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for LEFT. All error bars are 95% CIs.

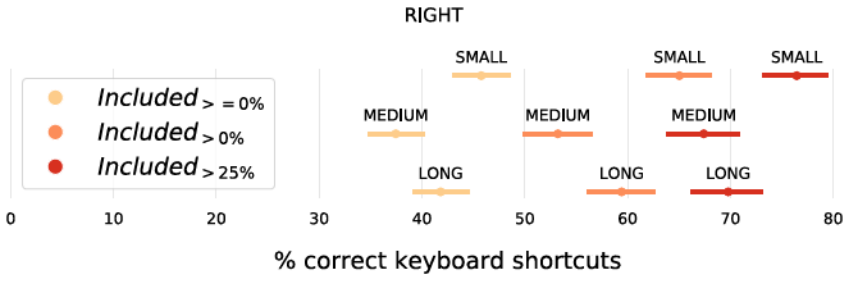


Figure i.16: Mean rate of *correct keyboard shortcuts* per layout for each threshold for RIGHT. All error bars are 95% CIs.

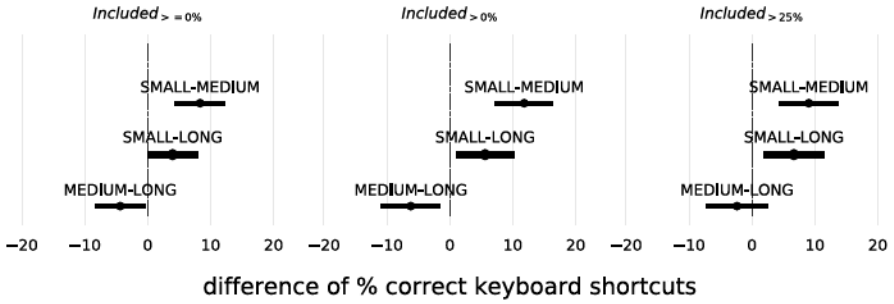


Figure i.17: Mean rate difference of *correct keyboard shortcuts* between menu item lengths for each threshold for RIGHT. All error bars are 95% CIs.

Table I.1: Set 1

Category	Item	Keyboard Shortcut	Keyboard Position
Vêtements	Manteau	Ctrl+O	R
Bureau	Enveloppe	Ctrl+P	R
Vêtements	Bouton	Ctrl+B	R
Loisirs	Flechettes	Ctrl+L	R
Animaux	Manchot	Shift+N	R
Vêtements	Chandail	Shift+A	L
Fruits	Cerise	Ctrl+C	L
Animaux	Grenouille	Ctrl+Q	L
Fruits	Nectarine	Shift+E	L
Bureau	Poubelle	Shift+Y	R
Bureau	Imprimante	Shift+I	R
Fruits	Fraise	Ctrl+U	R
Loisirs	Hockey	Shift+X	L
Légumes	Salade	Shift+S	L
Légumes	Champignon	Shift+W	L
Légumes	Persil	Ctrl+R	L
Animaux	Dinosaure	Ctrl+D	L
Loisirs	Karate	Shift+K	R

I.4 COMMAND – KEYBOARD SHORTCUT MAPPING MATERIAL

Tables I.1 to I.3 shows the 3 command-keyboard shortcut mappings that are used in Experiment 3.

Table i.2: Set 2

Category	Item	Keyboard Shortcut	Keyboard Position
Bureau	Enveloppe	Ctrl+E	L
Vêtements	Bouton	Shift+A	L
Vêtements	Manteau	Ctrl+T	L
Bureau	Poubelle	Shift+U	R
Vêtements	Chandail	Shift+C	L
Légumes	Salade	Ctrl+X	L
Fruits	Nectarine	Shift+N	R
Loisirs	Karate	Ctrl+S	L
Fruits	Cerise	Shift+Y	R
Loisirs	Hockey	Shift+O	R
Animaux	Dinosaure	Shift+K	R
Loisirs	Flechettes	Ctrl+F	L
Légumes	Champignon	Shift+G	L
Bureau	Imprimante	Ctrl+B	R
Animaux	Grenouille	Ctrl+I	R
Fruits	Fraise	Ctrl+R	L
Légumes	Persil	Ctrl+P	R
Animaux	Manchot	Shift+M	R

Table I.3: Set 3

Category	Item	Keyboard Shortcut	Keyboard Position
Vêtements	Chandail	Ctrl+S	L
Bureau	Imprimante	Ctrl+N	R
Bureau	Poubelle	Shift+P	R
Légumes	Champignon	Shift+C	L
Vêtements	Manteau	Ctrl+M	R
Fruits	Cerise	Shift+R	L
Vêtements	Bouton	Shift+U	R
Fruits	Fraise	Ctrl+F	L
Fruits	Nectarine	Ctrl+B	R
Légumes	Salade	Ctrl+L	R
Bureau	Enveloppe	Shift+K	R
Loisirs	Karate	Ctrl+T	L
Loisirs	Hockey	Shift+H	R
Animaux	Manchot	Ctrl+E	L
Loisirs	Flechettes	Shift+X	L
Animaux	Grenouille	Ctrl+G	L
Légumes	Persil	Shift+O	R
Animaux	Dinosaure	Shift+A	L

j APPENDIX TRANSITION

J.1 ADDITIONAL ANALYSIS OF MENU ITEM LENGTH – EXPERIMENT 2

J.1.1 BASELINE

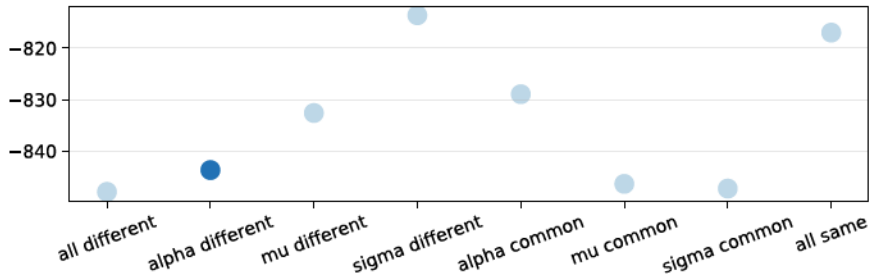


Figure j.1: AIC analysis indicates that the most suitable model is the one where the α value is different between the models

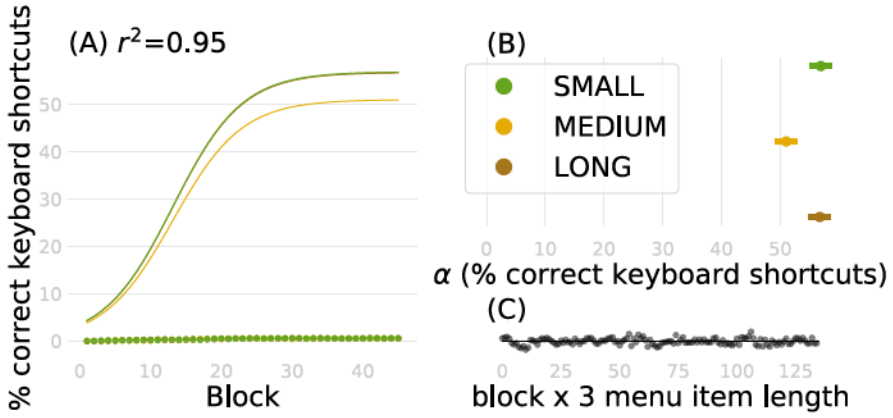


Figure j.2: (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot.

Table j.1

	menu item length	mean	lower	upper
0	LONG	56.73	1.86	1.90
1	MEDIUM	51.01	1.83	1.86
2	SMALL	56.91	1.87	1.91

j.1.2 RIGHT

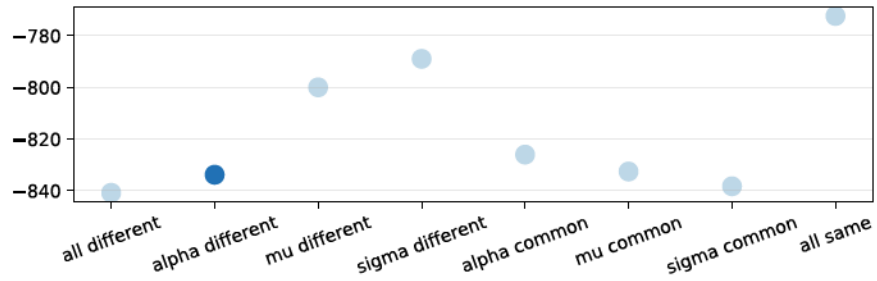


Figure j.3: AIC analysis indicates that the most suitable model is the one where the α value is different between the models

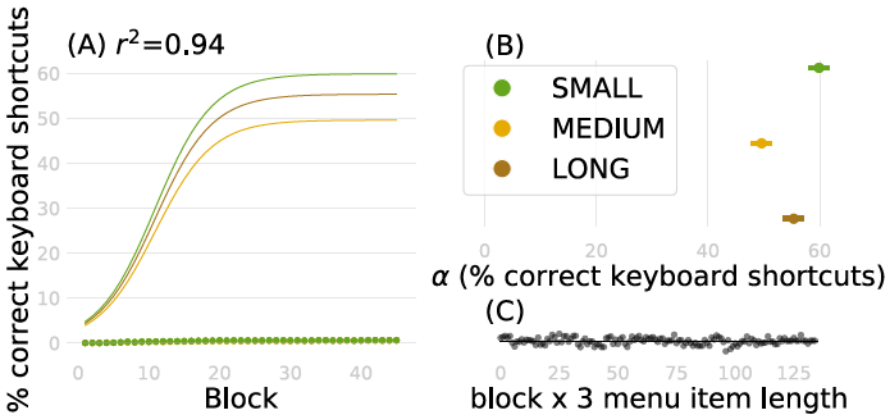


Figure j.4: (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot.

Table j.2

	menu item length	mean	lower	upper
0	LONG	55.36	1.79	1.82
1	MEDIUM	49.60	1.78	1.81
2	SMALL	59.89	1.81	1.84

J.2 ADDITIONAL ANALYSIS OF MENU LAYOUT – EXPERIMENT 3

J.2.1 FAMILIAR

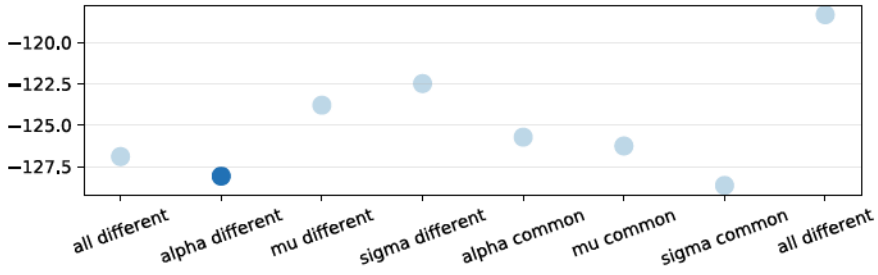


Figure j.5: AIC analysis indicates that the most suitable model is the one where the α value is different between the models

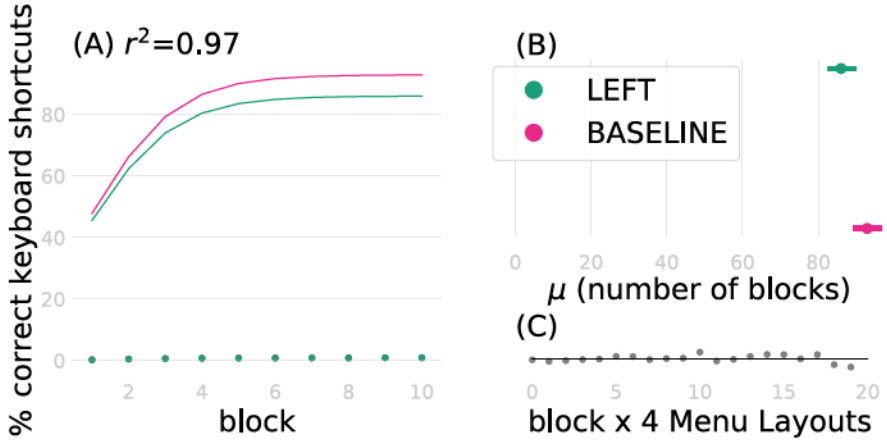


Figure j.6: (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot.

Table j.3

	layout	mean	lower	upper
2	BASELINE	92.90	3.59	3.85
3	LEFT	86.01	3.53	3.77

Table j.4

	layout	mean	lower	upper
2	BASELINE	75.77	4.77	5.67
3	LEFT	84.26	4.96	5.99

J.2.2 NOT-FAMILIAR

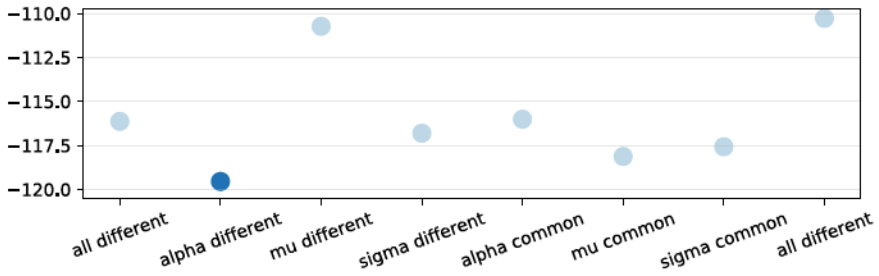


Figure j.7: AIC analysis indicates that the most suitable model is the one where the α value is different between the models

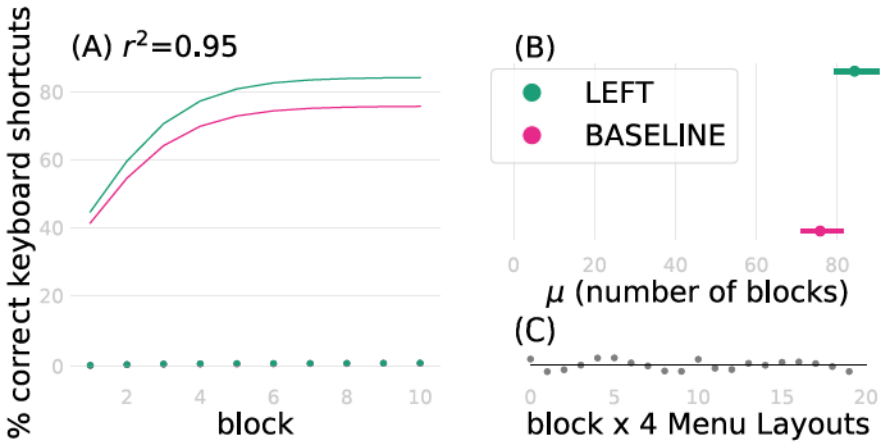


Figure j.8: (A) Regression results for each command - KS mapping quality. (B) 95% C.I. for α for each command - KS mapping quality. (C) Residual plot.

Table j.5

	menu_item_length	mean	lower_plt	upper_plt
0	LONG	86.35	3.81	4.10
1	MEDIUM	85.44	3.80	4.07
2	SMALL	77.82	3.75	4.00

J.3 ADDITIONAL ANALYSIS OF MENU ITEM LENGTH – EXPERIMENT 3

J.3.1 BASELINE

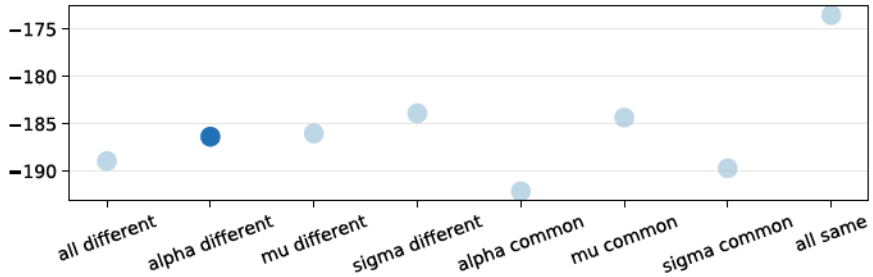


Figure j.9: AIC analysis indicates that the most suitable model is the one where the α value is different between the models

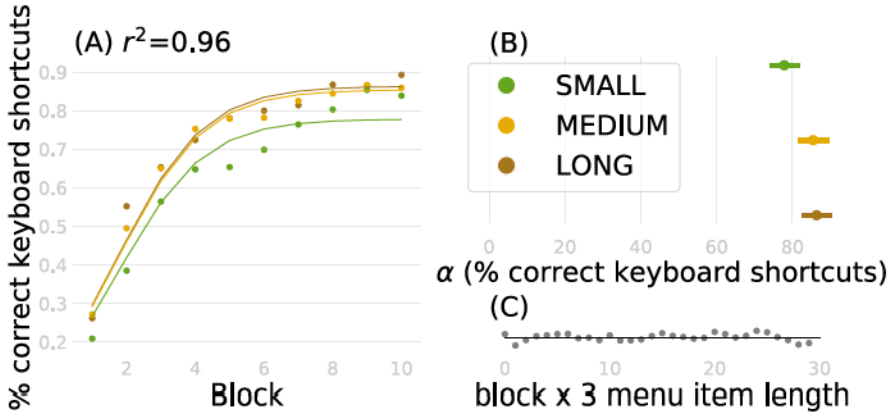


Figure j.10: (A) Regression results for each menu item length. (B) since the model. (C) Residual plot.

J.3.2 LEFT

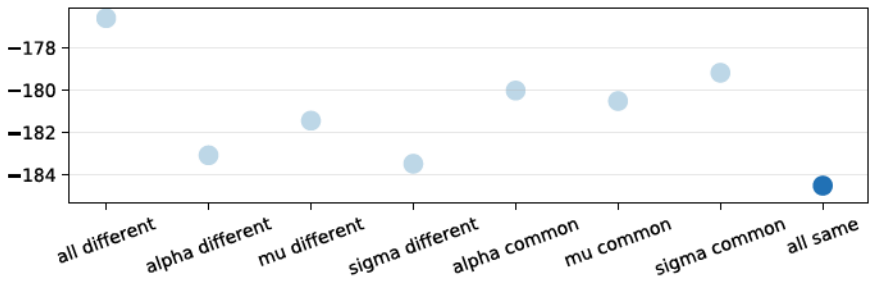


Figure J.11: AIC analysis indicates that the most suitable model is the one where the α value is different between the models

J.3.3 Time signal encoder-decoder with a 1D convolutional layer

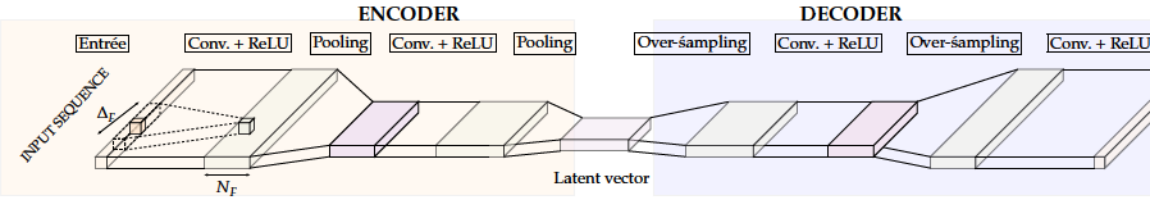


Figure J.12: Encoder-decoder of time-series from convolutional neural networks.

J.3.3.1 *Problem formulation and related work*

Automatic annotation can be formulated as a detection problem (segmentation + recognition) consisting, from a time series with two discrete variables (strategy and precision), in determining the phase of each moment among the three phases: Before, During and After the transition.

For this type of problem, several solutions have been proposed in the literature, including hidden Markov chains (HMM) [93, 132], change point detection algorithms [4], methods of label transfer following Dynamic Time Warping (DTW) alignment [60] or methods using deep neural networks [100]. While change point detection algorithms are completely unsupervised and based on break detection, label transfer methods implement an alignment between two sequences to transfer labels from one sequence to the other, which is not very useful for our problem where many labelled sequences are available.

Given the performance gains of deep neural networks for the labelling of time sequences, we have chosen to use them and more specifically, an encoder-decoder with a 1D convolutional layer [100].

J.3.3.2 Phase extraction: the encoder-decoder

Each sequence, coded at each instance by two values (strategy and accuracy), constitutes the input of the encoder/decoder. The output will have the same size as the input sequence (same number of points) but each moment will be represented by a single value encoding the phase: -1 for *Before the transition*, 0 for *During the transition* and 1 for *After the transition*.
The purpose of learning is to teach the network to decode the phase from the input sequence.

Figure j.12 summarizes the overall operation of the encoder/decoder by time convolutional neural network.

The purpose of the encoder is to encode the useful part of the information in a small latent vector. It consists of two convolutional layers [1] composed of N_F size filters Δ_F, followed by an activation function RELU (rectified linear unit) and a max-pooling layer performing subsampling while maintaining the maximum value on a window. These three layers can be repeated N times before obtaining the latent vector.

The purpose of the decoder is to create the desired output from the latent vector. It is composed of an oversampling layer followed by a convolutional layer and a RELU layer. As with the encoder, these three layers can be repeated N times. A softmax layer that transforms the output into the probability of belonging to the phases terminates the network.

Learning consists in learning the convolution filter coefficients that best predict phase sequences from input sequences on a learning basis. We used N=2, $N_F = (6.12)$ and $\Delta_F = 5$. Indeed, given the length of the signals (between 12 and 144) and the size of the descriptor (2), these values are a good compromise between a signal that is neither too summarized nor too detailed so as to clearly show the transitions.

Once the network is learned, it can associate to any input sequence $S_k = \{s_1, s_2..., s_{M_k}\}$ the output sequence $P_k = \{p_1, p_2..., p_{M_k}\}$ where $\forall i = 1...M_k, p_i \in \{-1, 0, 1\}$ corresponds to the phases *Before*, *During* and *After* the transition.

J.3.3.3 Estimation of the beginning and end of the transition

Ideally, the P_k sequence is composed of a single pass between -1 and 0 and a single pass between 0 and 1 directly providing the beginning and end of the desired transition. As this is not always the case, an additional step is necessary. It is based on simple heuristics:

- The beginning is the first passage between -1 and 0.
- The end is the first passage between 0 and 1.

1 Convolution is a multiplicative term-to-term operator with a sliding window that makes it possible to extract a local dependency descriptor while preserving the temporal relationships between the instances

These instances are refined so that the beginning of the transition corresponds to a selection that is not a menu only and the end of the transition corresponds to an error-free keyboard shortcut.

BIBLIOGRAPHY

[1] David Ahlstroem, Rainer Alexandrowicz, and Martin Hitz. "Improving menu interaction: a comparison of standard, force enhanced and jumping menus." In: *Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems* 1 (2006), pp. 1067–1076. DOI: 10.1145/1124772.1124932. URL: http://delivery.acm.org/10.1145/1130000/1124932/p1067-ahlstroem.pdf?ip=85.69.181.156{\&}id=1124932{\&}acc=ACTIVESERVICE{\&}key=7EBF6E77E86B478F.2E41AE6F08F798A5.FACEA2CAB50F059F.4D4702B0C3E38B35{\&}{_}{_}acm{_}{_}=1527449224{_}7e94fd7f08a51cca7648b10690b13dcfhttp://doi.acm.o.

[2] David Ahlström, Andy Cockburn, Carl Gutwin, and Pourang Irani. "Why it's quick to be square: Modelling new and existing hierarchical menu designs." In: *Proceedings of the 28th SIGCHI Conference on Human Factors in Computing Systems* (2010), pp. 1371–1380. DOI: 10.1145/1753326.1753534.

[3] Douglas G Altman. *Practical statistics for medical research*. CRC press, 1990.

[4] Samaneh Aminikhanghahi and Diane J. Cook. "A survey of methods for time series change point detection." In: *Knowledge and Information Systems* 51.2 (2017), pp. 339–367.

[5] John R Anderson. *The adaptive character of thought*. Psychology Press, 2013.

[6] Jeromy Anglim. "Strategies in Skill Acquisition: Reconciling Continuous Models of the Learning Curve with Abrupt Strategy Shifts." PhD thesis. 2011.

[7] Caroline Appert and Shumin Zhai. "Using Strokes As Command Shortcuts: Cognitive Benefits and Toolkit Support." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 2289–2298. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1519052. URL: http://doi.acm.org/10.1145/1518701.1519052.

[8] Apple Cooporation. *Toolbars - Windows and Views - Human Interface Guidelines for macOS Apps*. 2018. URL: https://developer.apple.com/macos/human-interface-guidelines/windows-and-views/toolbars/ (visited on 05/26/2018).

[9] B´ Ardossy, B M Bolker, Robert J. Hijmans, Susan E. Cameron, Juan L. Parra, Peter G. Jones, and Andy Jarvis. "Ecological Models and Data in R." In: *International Journal of Climatology* 25.15 (2007), pp. 1965–1978. ISSN: <null>. DOI: 10.1016/j.gene.2017.11.027. URL: http://books.google.com/books?hl=en{\&}lr={\&}id=7ZdtfhGHoz8C{\&}oi=fnd{\&}pg=PP2{\&}dq=Ecological+Models+and+Data+in+R{\&}ots=aZwx8lkksC{\&}sig=r7LiufBUdrW{_}0t06xtTjy2JMeWU{\%}0Ahttp://books.google.com/books?hl=en{\&}lr={\&}id=ZvoibTTS9QwC{\&}oi=fnd{\&}pg=PR19{\&}dq=Introduct.

[10] James A Arnett and Seth S Labovitz. "Effect of physical layout in performance of the Trail Making Test." In: *Psychological Assessment* 7.2 (1995), p. 220.

[11] ArtLebedev Studios. *Optimus*. 2018. URL: http://www.artlebedev.com/optimus/ (visited on 05/26/2018).

[12] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. "Flower Menus: A New Type of Marking Menu with Large Menu Breadth, Within Groups and Efficient Expert Mode Memorization." In: *Proceedings of the Working Conference on Advanced Visual Interfaces*. AVI '08. Napoli, Italy: ACM, 2008, pp. 15–22. ISBN: 978-1-60558-141-5. DOI: 10.1145/1385569.1385575. URL: http://doi.acm.org/10.1145/1385569.1385575.

[13] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. "Visual Menu Techniques." In: *ACM Computing Surveys* 49.4 (2016), pp. 1–41. ISSN: 03600300. DOI: 10.1145/3002171. URL: http://dl.acm.org/citation.cfm?doid=3022634.3002171.

[14] Gilles Bailly and Sylvain Malacria. "MenuInspector: Outil pour l'analyse des menus et cas d'étude." In: *Proceedings of the 25th Conference on l'Interaction Homme-Machine*. ACM. 2013, p. 103.

[15] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. "MenuOptimizer: interactive optimization of menu systems." In: *ACM symposium on User interface software and technology*. 2013, pp. 331 –342. DOI: 10.1145/2501988.2502024>. URL: https://hal.sorbonne-universite.fr/hal-01528320.

[16] Gilles Bailly, Thomas Pietrzak, Jonathan Deber, Daniel Wigdor, Telekom Innovation Laboratories, and T U Berlin. "Métamorphe : Augmenting H otkey U sage with A ctuated K eys." In: (2013), pp. 563–572.

[17] Gilles Bailly, Thomas Pietrzak, Jonathan Deber, and Daniel J Wigdor. "Métamorphe: augmenting hotkey usage with actuated keys." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, pp. 563–572.

[18] Gilles Bailly, Antti Oulasvirta, Duncan P Brumby, and Andrew Howes. "Model of visual search and selection time in linear menus." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2014, pp. 3865–3874.

[19] Ram Bajpai and Himanshu Chaturvedi. "Evaluation of Inter-Rater Agreement and Inter-Rater Reliability for Observational Data : An Overview of Concepts and Methods." In: *Journal of the Indian Academy of Applied Psychology* 41.3 (2015), pp. 20–27.

[20] Chris I Baker, Carl R Olson, and Marlene Behrmann. "Role of attention and perceptual grouping in visual statistical learning." In: *Psychological Science* 15.7 (2004), pp. 460–466.

[21] Olivier Bau and Wendy E. Mackay. "OctoPocus: A Dynamic Guide for Learning Gesture-based Command Sets." In: *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*. UIST '08. Monterey, CA, USA: ACM, 2008, pp. 37–46. ISBN: 978-1-59593-975-3. DOI: 10.1145/1449715.1449724. URL: http://doi.acm.org/10.1145/1449715.1449724.

[22] Benjamin Berman and Juan Pablo Hourcade. "Keyboard-card menus: A new presentation of non-standard shortcuts." In: *Journal of Universal Computer Science* 20.7 (2014), pp. 986–1005. ISSN: 09486968.

[23] Paul J Besl and Neil D McKay. "Method for registration of 3-D shapes." In: *Robotics-DL tentative*. International Society for Optics and Photonics. 1992, pp. 586–606.

[24] William L Bewley, Teresa I Koherts, David Schrnit William, and L Verplank. "CHI'83 Proceedings Human Factors Testing in the Design of Xerox's 8010 " Star " Office Workstation." In: (1983). URL: http://www.ece.uvic.ca/{~}aalbu/CENG4122009/bewley83.pdf.

[25] Suresh K Bhavnani and Bonnie E. John. "From sufficient to efficient usage." In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*. 1997, pp. 91–98. ISBN: 0897918029. DOI: 10.1145/258549.258615. URL: http://papers.cumincad.org/data/works/att/127c.content.04750.pdfhttp://portal.acm.org/citation.cfm?doid=258549.258615.

[26] Suresh K Bhavnani, Bonnie E John, and Ulrich Flemming. "The Strategic Use of CAD: An Empirically Inspired, Theory-based Course." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1999, pp. 183–190. ISBN: 0-201-48559-1. DOI: 10.1145/302979.303036. URL: http://delivery.acm.org/10.1145/310000/303036/p183-bhavnani.pdf?ip=85.69.181.156{\&}id=303036{\&}acc=ACTIVESERVICE{\&}key=7EBF6E77E86B478F.2E41AE6F08F798A5.FACEA2CAB50F059F.4D4702B0C3E38B35{\&}

{_}{_}acm{_}{_}=1527453447{_}616aca4da6c69469210f64c1db8a2e16http://doi.acm.org/10.

[27] Florian Block, Hans Gellersen, and Nicolas Villar. "Touch-display Keyboards: Transforming Keyboards into Interactive Surfaces." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* January 2010 (2010), pp. 1145–1154. DOI: 10.1145/1753326.1753498. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2010/04/p1145-block.pdf.

[28] Ned Block. "Semantics, conceptual role." In: *The Routledge Encylopedia of Philosophy*. 1997.

[29] Benjamin M Bolker. *Ecological models and data in R*. Princeton University Press, 2008.

[30] Daniel Buschek, Bianka Roppelt, and Florian Alt. "Extending Keyboard Shortcuts with Arm and Wrist Rotation Gestures." In: (2018). DOI: 10.1145/3173574.3173595.

[31] Georg Buscher, Edward Cutrell, and Meredith Ringel Morris. "What do you see when you're surfing?: using eye tracking to predict salient regions of web pages." In: *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM. 2009, pp. 21–30.

[32] John M Carroll and Mary Beth Rosson. *Paradox of the Active User*. 1987. DOI: 10.1017/CBO9781107415324.004. arXiv: arXiv:1011.1669v3.

[33] Tony F Chan, Gene H Golub, and Randall J LeVeque. "Algorithms for computing the sample variance: Analysis and recommendations." In: *The American Statistician* 37.3 (1983), pp. 242–247.

[34] Xiuli Chen, Gilles Bailly, Duncan P Brumby, Antti Oulasvirta, and Andrew Howes. "The emergence of interactive behavior: A model of rational menu search." In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 4217–4226.

[35] Marvin M Chun and Yuhong Jiang. "Contextual cueing: Implicit learning and memory of visual context guides spatial attention." In: *Cognitive psychology* 36.1 (1998), pp. 28–71.

[36] Andy Cockburn and Andrew Gin. "Faster cascading menu selections with enlarged activation areas." In: *Proceedings of the 2006 Conference on Graphics Interface* (2006), pp. 65–71. ISSN: 07135424. URL: https://ir.canterbury.ac.nz/bitstream/handle/10092/171/12602895{_}Main.pdf?sequence=1http://portal.acm.org/citation.cfm?id=1143079.1143091.

[37] Andy Cockburn, Carl Gutwin, and Alan Dix. "Hark no more: on the preregistration of chi experiments." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM. 2018, p. 141.

[38] Andy Cockburn, Carl Gutwin, and Saul Greenberg. "A Predictive Model of Menu Performance." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: ACM, 2007, pp. 627–636. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240723. URL: http://doi.acm.org/10.1145/1240624.1240723.

[39] Andy Cockburn, Carl Gutwin, Joey Scarr, and Sylvain Malacria. "Supporting Novice to Expert Transitions in User Interfaces." In: 47.2 (2014), pp. 1–36.

[40] Carlo Colombo, Alberto Del Bimbo, and Pietro Pala. "Semantics in visual information retrieval." In: *Ieee Multimedia* 6.3 (1999), pp. 38–53.

[41] Christopher M Conway and Morten H Christiansen. "Statistical learning within and between modalities: Pitting abstract against stimulus-specific representations." In: *Psychological science* 17.10 (2006), pp. 905–912.

[42] Mary M Crossan, Henry W Lane, and Roderick E White. "An organizational learning framework: From intuition to institution." In: *Academy of management review* 24.3 (1999), pp. 522–537.

[43] Jesus Dapena. "The evolution of high jumping technique: biomechanical analysis." In: *ISBS-Conference Proceedings Archive*. Vol. 1. 1. 2002.

[44] Sabiha S Daudi. "Environmental literacy: A system of best-fit for promoting environmental awareness in low literate communities." In: *Applied Environmental education and communication* 7.3 (2008), pp. 76–82.

[45] Matjaz Debevc, Beth Meyer, Dali Donlagic, and Rajko Svecko. "Design and Evaluation of an Adaptive Icon Toolbar." In: *User Modeling and User-Adapted Interaction* 6.9 (1996), pp. 1–21. URL: https://link.springer.com/content/pdf/10.1007/BF00126652.pdf.

[46] Peter F Delaney, Lynne M Reder, James J Staszewski, and Frank E Ritter. "The strategy-specific nature of improvement: The power law applies by strategy within task." In: *Psychological science* 9.1 (1998), pp. 1–7.

[47] Marc Destefano and Wayne D. Gray. "Where should researchers look for strategy discoveries during the acquisition of complex task performance? The case of space fortress." In: *Proceedings of the 38th Annual Conference of the Cognitive Science Society* (2016),

pp. 668–673. URL: http://homepages.rpi.edu/{~}grayw/pubs/papers/2016/marc16csc.pdf{\%}0Ahttp://files/15/space{_}fort{_}human{_}taskacq.pdf.

[48] R. F. Dillon, Jeff D. Edey, and Jo W. Tombaugh. "Measuring the True Cost of Command Selection: Techniques and Results." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '90. Seattle, Washington, USA: ACM, 1990, pp. 19–26. ISBN: 0-201-50932-6. DOI: 10.1145/97243.97247. URL: http://doi.acm.org/10.1145/97243.97247.

[49] Tom Djajadiningrat, Kees Overbeeke, and Stephan Wensveen. "But How, Donald, Tell Us How?: On the Creation of Meaning in Interaction Design Through Feedforward and Inherent Feedback." In: *Proceedings of the 4th Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*. DIS '02. London, England: ACM, 2002, pp. 285–291. ISBN: 1-58113-515-7. DOI: 10.1145/778712.778752. URL: http://doi.acm.org/10.1145/778712.778752.

[50] Pierre Dragicevic, Fanny Chevalier, and Stéphane Huot. "Running an hci experiment in multiple parallel universes." In: *CHI'14 Extended Abstracts on Human Factors in Computing Systems*. ACM. 2014, pp. 607–618.

[51] *Electron: Build cross platform desktop apps with JavaScript, HTML, and CSS*. https://electronjs.org/, Accessed: 2018-08-30. 2018.

[52] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[53] Paul Morris Fitts and Michael I Posner. "Human performance." In: (1967).

[54] James D. Foley and James D. Foley. *Computer graphics : principles and practice*. Addison-Wesley, 1990, p. 1174. ISBN: 0201121107. URL: https://books.google.fr/books/about/Computer{_}Graphics.html?id=-4ngT05gmAQC{\&}redir{_}esc=y.

[55] Jérémie Francone, Gilles Bailly, Eric Lecolinet, Nadine Mandran, and Laurence Nigay. "Wavelet Menus on Handheld Devices: Stacking Metaphor for Novice Mode and Eyes-free Selection for Expert Mode." In: *Proceedings of the International Conference on Advanced Visual Interfaces*. AVI '10. Roma, Italy: ACM, 2010, pp. 173–180. ISBN: 978-1-4503-0076-6. DOI: 10.1145/1842993.1843025. URL: http://doi.acm.org/10.1145/1842993.1843025.

[56] Qiufang Fu, Guangyu Bin, Zoltan Dienes, Xiaolan Fu, and Xiaorong Gao. "Learning without consciously knowing: Evidence from event-related potentials in sequence learning." In: *Consciousness and Cognition* 22.1 (2013), pp. 22–34. ISSN: 10902376. DOI: 10.1016/j.concog.2012.10.008. URL: https://www.sciencedirect.com/science/article/pii/S1053810012002231.

[57] Wai-Tat Fu and Wayne D Gray. "Memory versus perceptual-motor tradeoffs in a blocks world task." In: *Proceedings of the Twenty-second Annual conference of the Cognitive Science Society*. Erlbaum Hillsdale, NJ. 2000, pp. 154–159.

[58] Wai-Tat Fu and Wayne D. Gray. "Resolving the paradox of the active user: stable suboptimal performance in interactive tasks." In: *Cognitive Science* 28.6 (2004), pp. 901–935. ISSN: 1551-6709. DOI: 10.1207/s15516709cog2806_2. URL: http://dx.doi.org/10.1207/s15516709cog2806_2.

[59] Ovande Jr. Furtado and Jere D Gallagher. "Collecting inter and intra-rater reliability for the Furtado-Gallagher Computerized Observational Movement Pattern Assessment System." In: (2017). ISSN: 1558688X. DOI: 10.1177/0031512518769205.

[60] Y. Gao, S. Vedula, G. Lee, M. Lee, S. Khudanpur, and G. Hager. "Unsupervised surgical data alignment with application to automatic activity annotation." In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 4158–4163.

[61] Andrea L Gebhart, Elissa L Newport, and Richard N Aslin. "Statistical learning of adjacent and nonadjacent dependencies among nonlinguistic sounds." In: *Psychonomic bulletin & review* 16.3 (2009), pp. 486–490.

[62] *Github: Jquery Context Menu*. https://swisnl.github.io/jQuery-contextMenu/demo.html. Accessed: 2018-08-30. 2018.

[63] D Gittins. "Icon-based Human-computer Interaction." In: *Int. J. Man-Mach. Stud.* 24.6 (June 1986), pp. 519–543. ISSN: 0020-7373. DOI: 10.1016/S0020-7373(86)80007-4. URL: http://dx.doi.org/10.1016/S0020-7373(86)80007-4.

[64] Fernand Gobet, Peter CR Lane, Steve Croker, Peter CH Cheng, Gary Jones, Iain Oliver, and Julian M Pine. "Chunking mechanisms in human learning." In: *Trends in cognitive sciences* 5.6 (2001), pp. 236–243.

[65] Alison Gopnik and Henry M Wellman. "Reconstructing constructivism: Causal models, Bayesian learning mechanisms, and the theory theory." In: *Psychological bulletin* 138.6 (2012), p. 1085.

[66] Annabelle Goujon, André Didierjean, and Evelyne Marmeche. "Semantic contextual cuing and visual attention." In: *Journal of Experimental Psychology: Human Perception and Performance* 35.1 (2009), p. 50.

[67] Annabelle Goujon, André Didierjean, and Sarah Poulet. "The emergence of explicit knowledge from implicit learning." In: *Memory & cognition* 42.2 (2014), pp. 225–236.

[68] Annabelle Goujon, Andre Didierjean, and Simon Thorpe. "Investigating implicit statistical learning mechanisms through contextual cueing." In: *Trends in Cognitive Sciences* 19.9 (2015), pp. 524–533.

[69] Wayne D. Gray and Deborah A. Boehm-Davis. "Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior." In: *Journal of Experimental Psychology: Applied* 6.4 (2000), pp. 322–335. ISSN: 1076898X. DOI: 10.1037/1076-898X.6.4.322.

[70] Wayne D. Gray and Wai-Tat Fu. "Ignoring Perfect Knowledge In-the-world for Imperfect Knowledge In-the-head." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '01. Seattle, Washington, USA: ACM, 2001, pp. 112–119. ISBN: 1-58113-327-8. DOI: 10.1145/365024.365061. URL: http://doi.acm.org/10.1145/365024.365061.

[71] Wayne D Gray and Wai-Tat Fu. "Soft constraints in interactive behavior: The case of ignoring perfect knowledge in-the-world for imperfect knowledge in-the-head." In: *Cognitive Science* 28.3 (2004), pp. 359–382.

[72] Wayne D. Gray and John K. Lindstedt. "Plateaus, Dips, and Leaps: Where to Look for Inventions and Discoveries During Skilled Performance." In: *Cognitive Science* (2016), n/a–n/a. ISSN: 1551-6709. DOI: 10.1111/cogs.12412. URL: http://dx.doi.org/10.1111/cogs.12412.

[73] Wayne D. Gray, Chris R. Sims, Wai Tat Fu, and Michael J. Schoelles. "The soft constraints hypothesis: A rational analysis approach to resource allocation for interactive behavior." In: *Psychological Review* 113.3 (2006), pp. 461–482. ISSN: 0033295X. DOI: 10.1037/0033-295X.113.3.461.

[74] T. R G Green. "The necessity of syntax markers: Two experiments with artificial languages." In: *Journal of Verbal Learning and Verbal Behavior* 18.4 (1979), pp. 481–496. ISSN: 00225371. DOI: 10.1016/S0022-5371(79)90264-0.

[75] Thomas R. G. Green and Stephen J. Payne. "Organization and learnability in computer languages." In: *International Journal of Man-Machine Studies* 21.1 (1984), pp. 7–18.

[76] Mark Greenberg and Gilbert Harman. "Conceptual role semantics." In: (2006).

[77] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. "Strategies for Accelerating On-line Learning of Hotkeys." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '07. San Jose, California, USA: ACM, 2007, pp. 1591–1600. ISBN: 978-1-59593-593-9. DOI: 10.1145/1240624.1240865. URL: http://doi.acm.org/10.1145/1240624.1240865.

[78] Jonathan Grudin. "The case against user interface consistency." In: *Communications of the ACM* 32.10 (1989), pp. 1164–1173.

[79] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. "Faster Command Selection on Tablets with FastTap." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: ACM, 2014, pp. 2617–2626. ISBN: 978-1-4503-2473-1. DOI: 10.1145/2556288.2557136. URL: http://doi.acm.org/10.1145/2556288.2557136.

[80] Andrew Heathcote, Scott Brown, and D JK Mewhort. "The power law repealed: The case for an exponential law of practice." In: *Psychonomic bulletin & review* 7.2 (2000), pp. 185–207.

[81] *Hopper disassembler: http://hopperapp.com*. URL: http://hopperapp.com.

[82] Shuhua Hu. "Akaike information criterion." In: *Center for Research in Scientific Computation* 93 (2007).

[83] Daniel P Huttenlocher, Gregory A Klanderman, and William J Rucklidge. "Comparing images using the Hausdorff distance." In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 15.9 (1993), pp. 850–863.

[84] Apple Inc. *Menu Anatomy - Menus - macOS - Human Interface Guidelines - Apple Developer*. URL: https://developer.apple.com/design/human-interface-guidelines/macos/menus/menu-anatomy/.

[85] Arthur M Jacobs and Jonathan Grainger. "Models of visual word recognition: sampling the state of the art." In: *Journal of Experimental Psychology: Human perception and performance* 20.6 (1994), p. 1311.

[86] Mikkel Rønne Jakobsen and Kasper Hornæk. "Transient visualizations." In: *Proceedings of the 2007 conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction: design: activities, artifacts and environments - OZCHI '07*. 2007, p. 69. ISBN: 9781595938725. DOI: 10.1145/1324892.1324905. URL: http://mikkelrj.dk/wp-content/jakobsen2007-p69--76.pdfhttp://portal.acm.org/citation.cfm?doid=1324892.1324905.

[87] Elmar Kal, Rens Prosé, Marinus Winters, and John van der Kamp. "Does implicit motor learning lead to greater automatization of motor skills compared to explicit motor learning? A systematic review." In: (2018). DOI: 10.1371/journal.pone.0203591. URL: https://doi.org/10.1371/journal.pone.0203591.g001.

[88] *Keyboard - User Interaction - Human Interface Guidelines for macOS Apps*. URL: https://developer.apple.com/macos/human-interface-guidelines/user-interaction/keyboard/ (visited on 04/25/2018).

[89] Jong Wook Kim and Frank E. Ritter. "Learning, forgetting, and relearning for keystroke- and mouse-driven tasks: Relearning is important." In: *Human-Computer Interaction* 30.1 (2015), pp. 1–33. ISSN: 07370024. DOI: 10.1080/07370024.2013.828564.

[90] B Klampfer, R Flin, RL Helmreich, R Hausler, B Sexton, G Fletcher, P Field, St Staender, K Lauche, P Dieckmann, et al. "Enhancing performance in high risk environments: recommendations for the use of behavioural markers." In: *Ladenburg: Daimler-Benz Shiftung* (2001).

[91] Melanie Kleynen, Susy M. Braun, Michel H. Bleijlevens, Monique A. Lexis, Sascha M. Rasquin, Jos Halfens, Mark R. Wilson, Anna J. Beurskens, and Rich S. W. Masters. "Using a Delphi Technique to Seek Consensus Regarding Definitions, Descriptions and Classification of Terms Related to Implicit and Explicit Forms of Motor Learning." In: *PLOS ONE* 9.6 (June 2014), pp. 1–11. DOI: 10.1371/journal.pone.0100227. URL: https://doi.org/10.1371/journal.pone.0100227.

[92] Ron Kohavi et al. "A study of cross-validation and bootstrap for accuracy estimation and model selection." In: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, pp. 1137–1145.

[93] Sanjay Krishnan, Animesh Garg, Sachin Patil, Colin Lea, Gregory Hager, Pieter Abbeel, and Ken Goldberg. "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning." In: *The International Journal of Robotics Research* 36.13-14 (2017), pp. 1595–1618.

[94] Brian Krisler and Richard Alterman. "Training Towards Mastery : Overcoming the Active User Paradox." In: (2008), pp. 18–22.

[95] G. Kurtenbach, T. P. Moran, and W. Buxton. "Contextual Animation of Gestural Commands." In: *Computer Graphics Forum* 13.5 (1994), pp. 305–314. ISSN: 14678659. DOI: 10.1111/1467-8659.1350305.

[96] Gordon Paul Kurtenbach. "The design and evaluation of marking menus." PhD thesis. University of Toronto, 1993.

[97] Benjamin Lafreniere, Carl Gutwin, and Andy Cockburn. "Investigating the post-training persistence of expert interaction techniques." In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 24.4 (2017), p. 29.

[98] J. Richard Landis and Gary G. Koch. "The Measurement of Observer Agreement for Categorical Data." In: *Biometrics* 33.1 (1977), p. 159. ISSN: 0006341X. DOI: 10.2307/2529310. arXiv: NIHMS150003. URL: http://www.jstor.org/stable/2529310?origin=crossref.

[99] David M Lane, H Albert Napier, S Camille Peres, and Aniko Sandor. "Hidden Costs of Graphical User Interfaces: Failure to Make the Transition from Menus and Icon Toolbars to Keyboard Shortcuts." In: *International Journal of Human-Computer Interaction* 18.2 (2005), pp. 133–144. ISSN: 1044-7318. DOI: 10.1207/s15327590ijhc1802_1. URL: http://www.tandfonline.com/doi/abs/10.1207/s15327590ijhc1802{_}1.

[100] Colin Lea, René Vidal, Austin Reiter, and Gregory D. Hager. "Temporal Convolutional Networks: A Unified Approach to Action Segmentation." In: 2016, pp. 47–54.

[101] Yang Li, Samy Bengio, and Gilles Bailly. "Predicting Human Performance in Vertical Menu Selection Using Deep Learning." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM. 2018, p. 29.

[102] Rungtai Lin. "A study of visual features for icon design." In: *Design Studies* 15.2 (1994), pp. 185–197. ISSN: 0142694X. DOI: 10.1016/0142-694X(94)90024-8.

[103] Wanyu Liu, Gilles Bailly, and Andrew Howes. "Effects of Frequency Distribution on Linear Menu Performance." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM. 2017, pp. 1307–1312.

[104] Kenneth Lodding. "Iconic Interfacing." In: *IEEE Comput. Graph. Appl.* 3.2 (Feb. 1983), pp. 11–20. ISSN: 0272-1716. DOI: 10.1109/MCG.1983.262982. URL: http://dx.doi.org/10.1109/MCG.1983.262982.

[105] Abraham S. Luchins. "Mechanization in problem solving: The effect of Einstellung." In: *Psychological Monographs* 54.6 (1942), pp. i–95. ISSN: 0096-9753. DOI: 10.1037/h0093502. URL: http://doi.apa.org/getdoi.cfm?doi=10.1037/h0093502.

[106] Xiaoyue Ma, Nada Matta, Jean-Pierre Cahier, Chunxiu Qin, and Yanjie Cheng. "From action icon to knowledge icon: Objective-oriented icon taxonomy in computer science." In: *Displays* 39 (2015), pp. 68–79.

[107] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. "Promoting Hotkey Use Through Rehearsal with ExposeHK." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, pp. 573–582. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2470735. URL: http://doi.acm.org/10.1145/2470654.2470735.

[108] Sylvain Malacria, Joey Scarr, Andy Cockburn, Carl Gutwin, and Tovi Grossman. "Skillometers: Reflective Widgets That Motivate and Help Users to Improve Performance." In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. St. Andrews, Scotland, United Kingdom: ACM, 2013, pp. 321–330. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2501996. URL: http://doi.acm.org/10.1145/2501988.2501996.

[109] Siné McDougall, Victoria Tyrer, and Simon Folkard. "Searching for signs, symbols, and icons: effects of time of day, visual complexity, and grouping." In: *Journal of Experimental Psychology: Applied* 12.2 (2006), p. 118.

[110] Marry L. McHugh. "Interrater reliability: the kappa statistic." In: *Biochemia Medica* (2012), pp. 276–282. ISSN: 18467482. DOI: 10.11613/BM.2012.031. arXiv: 9809069v1 [arXiv:gr-qc]. URL: http://www.biochemia-medica.com/node/501.

[111] Microsoft Corporation. *Ribbons (Windows)*. URL: https://docs.microsoft.com/en-us/windows/desktop/uxguide/cmd-ribbons.

[112] Microsoft Corporation. *Microsoft Windows user experience*. 1999. URL: https://msdn.microsoft.com/en-us/library/ms971323.aspx{\#}atg{_}keyboardshortcuts{_}creating{_}shortcut{_}keys{_}and{_}access{_}keys.

[113] Microsoft Corporation. *Toolbars (Windows)*. 2018. URL: https://msdn.microsoft.com/en-us/library/windows/desktop/dn742395(v=vs.85).aspx (visited on 05/26/2018).

[114] A Naamad, DT Lee, and W-L Hsu. "On the maximum empty rectangle problem." In: *Discrete Applied Mathematics* 8.3 (1984), pp. 267–277.

[115] Carlos Nakamura and Qing Zeng-Treitler. "A taxonomy of representation strategies in iconic communication." In: *International journal of human-computer studies* 70.8 (2012), pp. 535–551.

[116] *Neuroscience For Kids*. URL: https://faculty.washington.edu/chudler/chmemory.html.

[117] A. Newell and P. S. Rosenbloom. "The Soar Papers (Vol. 1)." In: ed. by Paul S. Rosenbloom, John E. Laird, and Allen Newell. Cambridge, MA, USA: MIT Press, 1993. Chap. Mechanisms of Skill Acquisition and the Law of Practice, pp. 81–135. ISBN: 0-262-68071-8. URL: http://dl.acm.org/citation.cfm?id=162580.162586.

[118] Allen Newell and Paul S Rosenbloom. "Mechanisms of skill acquisition and the law of practice." In: *Cognitive skills and their acquisition* 1.1981 (1981), pp. 1–55.

[119] Donald A Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 1988.

[120] Richard C Omanson, Craig S Miller, Elizabeth Young, and David Schwantes. "Comparison of Mouse and Keyboard Efficiency Effects of Practice." In: (2010), pp. 600–604.

[121] *Oracle: Java Documentation Swing*. https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html,. Accessed: 2018-08-30. 2018.

[122] S Camille Peres, Franklin P. Tamborello, Michael D Fleetwood, Phillip Chung, and Danielle L. Paige-Smith. "Keyboard Shortcut Usage: The Roles of Social Factors and Computer Experience." In: *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 48.5 (2004), pp. 803–807. ISSN: 1541-9312. DOI: 10.1177/154193120404800513. URL: http://journals.sagepub.com/doi/10.1177/154193120404800513.

[123] S Camille Peres, Michael D Fleetwood, Minmin Yang, P Franklin, and Danielle Paige Smith. "Pros , Cons , and Changing Behavior : An Application in the Use of the Keyboard to Issue Commands." In: September (2005). DOI: 10.1177/154193120504900501.

[124] Pierre Perruchet and Sebastien Pacton. "Implicit learning and statistical learning: One phenomenon, two approaches." In: *Trends in cognitive sciences* 10.5 (2006), pp. 233–238.

[125] Pierre Perruchet and Sebastien Pacton. "Implicit learning and statistical learning: One phenomenon, two approaches." In: *Trends in cognitive sciences* 10.5 (2006), pp. 233–238.

[126] Pierre Perruchet and Annie Vinter. "PARSER: A model for word segmentation." In: *Journal of memory and language* 39.2 (1998), pp. 246–263.

[127] Thomas Pietrzak, Sylvain Malacria, and Gilles Bailly. "CtrlMouse Et TouchCtrl: Duplicating Mode Delimiters on the Mouse." In: *Proceedings of the 26th Conference on L'Interaction Homme-Machine*. IHM '14. Villeneuve d'Ascq, France: ACM, 2014, pp. 38–47. ISBN: 978-1-4503-2935-4. DOI: 10.1145/2670444.2670447. URL: http://doi.acm.org/10.1145/2670444.2670447.

[128] C. Powell, S.C. Peres, V. Nguyen, K.E. Bruton, and L. Muse. "Using the keyboard to issue commands: The relation of observing others using efficient techniques on the weightings of costs and benefits." In: *Proceedings of the Human Factors and Ergonomics Society* 3 (2009). ISSN: 10711813. DOI: 10.1518/107118109X12524444079398. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1024.8192{\&}rep=rep1{\&}type=pdf.

[129] James O Prochaska and Wayne F Velicer. "The transtheoretical model of health behavior change." In: *American journal of health promotion* 12.1 (1997), pp. 38–48.

[130] *Pycharm: Python IDE for Professional Developers*. https://www.jetbrains.com/pycharm/. Accessed: 2018-08-30. 2018.

[131] *Qt Software development made smarter*. qt.io. Accessed: 2018-08-30. 2018.

[132] L. Rabiner and B. Juang. "An introduction to hidden Markov models." In: *IEEE ASSP Magazine* 3.1 (1986), pp. 4–16. ISSN: 0740-7467. DOI: 10.1109/MASSP.1986.1165342.

[133] Roger W Remington, Ho Wang Holman Yuen, and Harold Pashler. "With practice, keyboard shortcuts become faster than menu selection: A crossover interaction." In: *Journal of Experimental Psychology: Applied* 22.1 (2016), pp. 95–106. ISSN: 1076898X. DOI: 10.1037/xap0000069. URL: http://doi.apa.org/getdoi.cfm?doi=10.1037/xap0000069.

[134] Stephen Rollnick, William R Miller, and Christopher Butler. *Motivational interviewing in health care: helping patients change behavior*. Guilford Press, 2008.

[135] Jarrett K Rosenberg. "Evaluating the suggestiveness of command names." In: *Behaviour & Information Technology* 1.4 (1982), pp. 371–400.

[136] Jarrett Rosenberg. "Lexical semantics in human-computer communication." In: *Proceedings of the 10th International Conference on Computational Linguistics and 22nd annual meeting on Association for Computational Linguistics*. Association for Computational Linguistics. 1984, pp. 428–431.

[137] Quentin Roy, Sylvain Malacria, Yves Guiard, Eric Lecolinet, and James Eagan. "Augmented Letters: Mnemonic Gesture-based Shortcuts." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '13. Paris, France: ACM, 2013, pp. 2325–2328. ISBN: 978-1-4503-1899-0. DOI: 10.1145/2470654.2481321. URL: http://doi.acm.org/10.1145/2470654.2481321.

[138] Edgar Rubin. "Synsoplevede figurer." In: (1915).

[139] Jose Moreno Rubio and Paul Janecek. "Floating Pie Menus : Enhancing the functionality of Contextual Tools." In: *Learning* (2002), pp. 39–40. URL: http://uist.hosting.acm.org/archive/adjunct/2002/pdf/demos/p39-rubio.pdfhttp://www.acm.org/uist/archive/adjunct/2002/pdf/demos/p39-rubio.pdf.

[140] Jenny R Saffran, Richard N Aslin, and Elissa L Newport. "Statistical learning by 8-month-old infants." In: *Science* 274.5294 (1996), pp. 1926–1928.

[141] Jenny R Saffran and Erik D Thiessen. "Domain-general learning capacities." In: *Blackwell handbook of language development* (2007), pp. 68–86.

[142] Jenny R Saffran, Elissa L Newport, Richard N Aslin, Rachel A Tunick, and Sandra Barrueco. "Incidental language learning: Listening (and learning) out of the corner of your ear." In: *Psychological science* 8.2 (1997), pp. 101–105.

[143] Joey Scarr, Andy Cockburn, Carl Gutwin, and Philip Quinn. "Dips and Ceilings: Understanding and Supporting Transitions to Expertise in User Interfaces." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: ACM, 2011, pp. 2741–2750. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979348. URL: http://doi.acm.org/10.1145/1978942.1979348.

[144] Richard A Schmidt, Douglas E Young, Stephan Swinnen, and Diane C Shapiro. "Summary knowledge of results for skill acquisition: Support for the guidance hypothesis." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 15.2 (1989), p. 352.

[145] Richard A Schmidt, Tim Lee, Carolee Winstein, Gabriele Wulf, and Howard Zelaznik. *Motor Control and Learning, 6E*. Human kinetics, 2018.

[146] Yilei Shi, Haimo Zhang, Hasitha Rajapakse, and Nuwan Tharaka Perera. "GestAKey : Touch Interaction on Individual Keycaps." In: (2018), pp. 1–12.

[147] Ben Shneiderman and Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. Vol. 215. 7. 2005, pp. 1–672. ISBN: 0321197860. DOI: 10.1038/sj.bdj.2013.932. URL: http://www.ncbi.nlm.nih.gov/pubmed/24502408.

[148] Lisa Silver. *Graphic Design that Works: Secrets for Successful Logo, Magazine, Brochure, Promotion and Identity Design*. Rockport, 2004.

[149] Herbert A. Simon. "Theories of Decision-Making in Economics and Behavioural Science." In: *Surveys of Economic Theory: Resource Allocation*. London: Palgrave Macmillan UK, 1966, pp. 1–28. ISBN: 978-1-349-00210-8. DOI: 10.1007/978-1-349-00210-8_1. URL: http://dx.doi.org/10.1007/978-1-349-00210-8_1.

[150] Herbert A Simon. *The sciences of the artificial*. MIT press, 1996.

[151] Linda Smith and Chen Yu. "Infants rapidly learn word-referent mappings via cross-situational statistics." In: *Cognition* 106.3 (2008), pp. 1558–1568.

[152] Andrea C Smyth and David R Shanks. "Awareness in contextual cuing with extended and concurrent explicit tests." In: *Memory & Cognition* 36.2 (2008), pp. 403–415.

[153] Andrea C Smyth and David R Shanks. "Awareness in contextual cuing with extended and concurrent explicit tests." In: *Memory & Cognition* 36.2 (2008), pp. 403–415.

[154] Barry M Staw. *The Escalation of Commitment To a Course of Action*. Tech. rep. 4. 1981, pp. 577–587. URL: https://www.gwern.net/docs/sunkcosts/1981-staw.pdf.

[155] Holly L Storkel. "Developmental differences in the effects of phonological, lexical and semantic variables on word learning by infants." In: *Journal of child language* 36.2 (2009), pp. 291–321.

[156] R. W. Swezey and E. G. Davis. "A Case Study of Human Factors Guidelines in Computer Graphics." In: *IEEE Computer Graphics and Applications* 3.8 (1983), pp. 21–30. ISSN: 0272-1716. DOI: 10.1109/MCG.1983.263294.

[157] Susanne Tak, Piet Westendorp, and Iris van Rooij. "Satisficing and the Use of Keyboard Shortcuts: Being Good Enough Is Enough?" In: *Interacting with computers* 25.5 (2013), pp. 404–416.

[158] Sussane Tak. "The Use of Keyboard Shortcuts Optimizing versus satisficing in the use of complex technology." In: *Management* 11 (2007).

[159] Wayne A Taylor. "Change-point analysis: a powerful new tool for detecting changes." In: *preprint, available as http://www. variation. com/cpa/tech/changepoint. html* (2000).

[160] Caitlin Tenison and John R. Anderson. "Modeling the distinct phases of skill acquisition." In: *Journal of Experimental Psychology: Learning Memory and Cognition* 42.5 (2016), pp. 749–767. ISSN: 02787393. DOI: 10.1037/xlm0000204.

[161] Caitlin Tenison, Jon M. Fincham, and John R. Anderson. "Phases of learning: How skill acquisition impacts cognitive processing." In: *Cognitive Psychology* 87 (2016), pp. 1 –28. ISSN: 0010-0285. DOI: http://dx.doi.org/10.1016/j.cogpsych.2016.03.001. URL: http://www.sciencedirect.com/science/article/pii/S0010028515300657.

[162] Caitlin Tenison and Christopher J. MacLellan. "Modeling strategy use in an intelligent tutoring system: Implications for strategic flexibility." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 8474 LNCS (2014), pp. 466–475. ISSN: 16113349. DOI: 10.1007/978-3-319-07221-0_58.

[163] Cyril Thomas, André Didierjean, François Maquestiaux, and Annabelle Goujon. "On the limits of statistical learning: Intertrial contextual cueing is confined to temporally close contingencies." In: *Attention, Perception, & Psychophysics* (2018), pp. 1–16.

[164] Edward V. Thomas. "Modelling Binary Data." In: *Technometrics* 35.2 (1993), pp. 224–224. DOI: 10.1080/00401706.1993.10485049. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00401706.1993.10485049. URL: https://www.tandfonline.com/doi/abs/10.1080/00401706.1993.10485049.

[165] William Homan Thorpe. "Learning and instinct in animals." In: (1956).

[166] Juan M Toro, Scott Sinnett, and Salvador Soto-Faraco. "Speech segmentation by statistical learning depends on attention." In: *Cognition* 97.2 (2005), B25–B34.

[167] Michael Tuck. *Gestalt Principles Applied in Design*. 2010.

[168] Nicholas B Turk-Browne. "Statistical learning in perception." In: *Encyclopedia of the Sciences of Learning*. Springer, 2012, pp. 3182–3185.

[169] Nicholas B Turk-Browne, Justin A Jungé, and Brian J Scholl. "The automaticity of visual statistical learning." In: *Journal of Experimental Psychology: General* 134.4 (2005), p. 552.

[170] Nicholas B Turk-Browne and Brian J Scholl. "Flexible visual statistical learning: transfer across space and time." In: *Journal of Experimental Psychology: Human Perception and Performance* 35.1 (2009), p. 195.

[171] Md Sami Uddin, Carl Gutwin, and Andy Cockburn. "The effects of artificial landmarks on learning and performance in spatial-memory interfaces." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM. 2017, pp. 3843–3855.

[172] Viswanath Venkatesh, Michael G Morris, Gordon B Davis, and Fred D Davis. "User Acceptance of Information Technology: Toward a Unified View." In: 27.3 (2003), pp. 425–478. DOI: doi.org/10.2307/30036540.

[173] Jo Vermeulen, Kris Luyten, Elise van den Hoven, and Karin Coninx. "Crossing the bridge over Norman's Gulf of Execution: revealing feedforward's true identity." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM. 2013, pp. 1931–1940.

[174] Nefs Walker and Judith Reitmun Olson. "Designing keybindings to be easy to learn and resistant to forgetting even when the set of commands is large." In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM. 1988, pp. 201–206.

[175] Jingtao Wang and John Canny. "FingerSense: augmenting expressiveness to physical pushing button by fingertip identification." In: *Conference on Human Factors in Computing Systems* January 2004 (2004), pp. 1267–1270. DOI: 10.1145/985921.986040. URL: http://portal.acm.org/citation.cfm?id=985921.986040.

[176] Felix A Wichmann and N Jeremy Hill. "The psychometric function:{I}. {Fitting}, sampling, and goodness of fit." In: 63.8 (2003), pp. 1–21.

[177] Felix A Wichmann and N Hill. "The psychometric function: II. Boostrap-based confidence intervals and sampling." In: *Perception & Psychophysics* 63.8 (2001), pp. 1314–1329.

[178] Windows-Sdk-Content. *Keyboard - Windows applications*. URL: https://docs.microsoft.com/en-us/windows/desktop/uxguide/inter-keyboard.

[179] Ian H. Witten, John G. Cleary, and Saul Greenberg. "On frequency-based menu-splitting algorithms." In: *International Journal of Man-Machine Studies* 21.2 (1984), pp. 135–148. ISSN: 0020-7373. DOI: 10.1016/S0020-7373(84)80063-2. URL: https://www.sciencedirect.com/science/article/pii/S0020737384800632.

[180] Ludwig Wittgenstein. *Philosophical investigations*. John Wiley & Sons, 2009.

[181] Tarah SA Wright. "Definitions and frameworks for environmental sustainability in higher education." In: *Higher education policy* 15.2 (2002), pp. 105–120.

[182] Sarah K. A. Wynton and Jeromy Anglim. "Abrupt strategy change underlies gradual performance change: Bayesian hierarchical models of component and aggregate strategy use." In: *Journal of Experimental Psychology: Learning, Memory, and Cognition* 43.10 (2017), pp. 1630–1642. ISSN: 1939-1285. DOI: 10.1037/xlm0000404. URL: https://doi.org/10.1037/xlm0000404http://doi.apa.org/getdoi.cfm?doi=10.1037/xlm0000404.

[183] Dahlia W Zaidel. "Worlds apart: Pictorial semantics in the left and right cerebral hemispheres." In: *Current Directions in Psychological Science* 3.1 (1994), pp. 5–8.

[184] Shumin Zhai. "Foundational Issues in Touch-Surface Stroke Gesture Design — An Integrative Review." In: *Foundations and Trends® in Human–Computer Interaction* 5.2 (2012), pp. 97–205. ISSN: 1551-3955. DOI: 10.1561/1100000012. URL: http://www.nowpublishers.com/article/Details/HCI-012.

[185] H Zhang and Y Li. "GestKeyboard: Enabling Gesture-Based Interaction on Ordinary Physical Keyboard." In: *Proceedings of CHI 2014* (2014), pp. 1675–1684. DOI: 10.1145/2556288.2557362.

[186] S Zhao, M Agrawala, and K Hinckley. "Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus." In: *Proceedings of the SIGCHI conference on Human Factors in computing systems* (2006), pp. 1077–1086. ISSN: 1062-9432. DOI: 10.1145/1124772.1124933.

[187] Shengdong Zhao and Ravin Balakrishnan. "Simple vs. Compound Mark Hierarchical Marking Menus." In: *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*. UIST '04. Santa Fe, NM, USA: ACM, 2004, pp. 33–42. ISBN: 1-58113-957-8. DOI: 10.1145/1029632.1029639. URL: http://doi.acm.org/10.1145/1029632.1029639.

[188] Jingjie Zheng and Daniel Vogel. "Finger-Aware Shortcuts." In: *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM. 2016, pp. 4274–4285.

[189] Jingjie Zheng, Blaine Lewis, Jeff Avery, and Daniel Vogel. "FingerArc and FingerChord." In: *The 31st Annual ACM Symposium on User Interface Software and Technology - UIST '18* (2018), pp. 347–363. DOI: 10.1145/3242587.3242589. URL: http://dl.acm.org/citation.cfm?doid=3242587.3242589.

[190] George Kingsley Zipf. *Human behavior and the principle of least effort: An introduction to human ecology*. Ravenio Books, 2016.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Favoriser et caractériser la transition des menus vers les raccourcis claviers

Mots clés : IHM, sélection de commandes,transition vers des méthodes expertes

Résumé : Les utilisateurs sélectionnent fréquemment des commandes sur des ordinateurs personnels, des tablettes ou des smartphones. Ils peuvent utiliser des méthodes novices comme les menus linéaires, les bandeaux de commandes ou les barres d'outils. Ils peuvent également utiliser des méthodes expertes telles que les raccourcis clavier ou les raccourcis gestuels qui sont plus efficaces. Cependant, nous observons que de nombreux utilisateurs continuent d'utiliser des méthodes novices, même des utilisateurs expérimentés, parce qu'ils ne parviennent pas à faire la transition de méthodes novices à des méthodes expertes. Cela peut avoir un impact important sur la productivité.
Dans cette thèse, j'étudie la transition des menus vers les raccourcis clavier sur les ordinateurs personnels. Je me concentre sur deux objectifs complémentaires.
Le premier objectif consiste à favoriser l'utilisation des raccourcis clavier en concevant de nouvelles techniques d'interaction. Mon approche consiste à (1) identifier les éléments clés d'une commande (nom, icône et raccourci), (2) décrire comment ces éléments sont représentés dans les interfaces traditionnelles et leurs impacts sur la performance et (3) proposer des solutions alternatives. En particulier, je conçois, réalise et évalue deux nouvelles techniques d'interaction qui explorent différents emplacements des raccourcis clavier : la première intègre les raccourcis clavier dans les icônes. La seconde explore la position des raccourcis clavier dans un item de menus. Nous montrons que les deux techniques d'interaction sont prometteuses pour promouvoir l'utilisation des raccourcis clavier. En outre, j'examine comment l'association ¡ nom de la commande - raccourci clavier¿ influence l'utilisation des raccourcis clavier.
Le deuxième objectif consiste à mieux caractériser la transition des menus vers les raccourcis clavier. Je souligne les incohérences entre les caractérisations théoriques et empiriques de cette transition, ce qui rend difficile l'évaluation et la comparaison des techniques d'interaction. Je présente une nouvelle méthodologie pour estimer des marqueurs comportementaux théoriques sur des données empiriques. En particulier, je conçois et évalue un algorithme pour identifier automatiquement le début et la fin de la transition. J'apporte également de nouvelles perspectives sur le comportement des utilisateurs avant, pendant et après la transition. Cette thèse offre (1) une meilleure compréhension de la transition des menus vers les raccourcis clavier et des facteurs impliqués dans cette transition, (2) de nouvelles méthodes pour caractériser cette transition et (3) deux nouvelles techniques d'interaction pour promouvoir les raccourcis clavier.

Title : Promoting and characterizing the menu to keyboard shortcuts transition

Keywords : HCI, command selection, transition to expert methods

Abstract : Users frequently select commands on personal computers, tablets or smartphones. They can use novice methods such as linear menus, ribbons and toolbars. They can also use expert methods such as keyboard shortcuts and stroke shortcuts that are more efficient. However, we observe that many users continue to use novice methods even experimented users because they fail to make the transition from novice to expert methods. This can have a strong impact on productivity.
In this thesis, I investigate the transition from menus to keyboard shortcuts on personal computers and focus on two complementary objectives. The first objective consists of promoting keyboard shortcut usage by designing novel interaction techniques. My approach consists of (1) identifying the key elements of a command (name, icon and shortcut), (2) describing how these elements are represented in traditional interfaces and their impact on performance and (3) proposing alternative designs. In particular, I design, implement and evaluate two novel interaction techniques exploring different locations of the keyboard shortcut cues: The first one blends the keyboard shortcut cues into command icons. The second one explores the relative position of the keyboard shortcut cues with their corresponding command labels. We show that both interaction techniques are promising for promoting keyboard shortcut usage. In addition, I investigate how the command name - keyboard shortcut mapping impacts keyboard shortcut usage.
The second objective consists of better characterizing the transition from menus to keyboard shortcuts. I highlight inconsistencies between theoretical and empirical characterizations of this transition making difficult to evaluate and compare interaction techniques. I present a novel methodology to estimate theoretical behavioral markers on empirical data. In particular, I design and evaluate an algorithm to automatically identify the beginning and the end of the transition. I also provide new insights regarding users behaviors before, during and after the transition. This thesis offers (1) a better understanding of the transition from menus to keyboard shortcuts and the design factors involved in this transition, (2) novel methods to characterize this transition and (3) two novel interaction techniques to promote keyboard shortcuts.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France