Modeling Goal-Directed User Exploration in Human-Computer Interaction

Leonghwee Teo

February 2011

CMU-HCII-11-102

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15123

Thesis Committee:
Bonnie E. John (Chair)
Aniket Kittur
Brad A. Myers
Peter L. Pirolli (Palo Alto Research Center)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This work was supported in part by IBM, NASA, Boeing, NEC, PARC, the Office of Naval Research (ONR) under contract number N00014-03-1-0086, and a Scholarship from DSO National Laboratories. The views and conclusions in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of IBM, NASA, Boeing, NEC, PARC, DSO, ONR, or the U.S. Government.

Keywords: CogTool-Explorer, CogTool, Predictive Human Performance Cognitive Modeling, Human Behavior Representation, Exploratory Behavior, Information Foraging, Information Scent, Hierarchical Visual Search, User-Interface Design, User-Interface Evaluation, SNIF-ACT, AutoCWW, ACT-R

for the benefit of all sentient beings

ABSTRACT

Designing user-interfaces so that first-time or infrequent users can accomplish their goals by exploration has been an enduring challenge in Human-Computer Interaction. Iterative user-testing is an effective but costly method to develop user-interfaces that support use through exploration. A complementary method is to use modeling tools that can generate predictions of user exploration given a user-interface and a goal description.

Recent computational models of goal-directed user exploration have focused on predicting user exploration of websites and demonstrated how predictions can inform user-interface design. These models employ the common concepts of label following and information scent: that the user's choice is partly determined by the semantic relevance between the user's goal and the options presented in the user-interface. However, in addition to information scent, other factors including the layout position and grouping of options in the user-interface also affect user exploration and the likelihood of success.

This dissertation contributes a new model of goal-directed user exploration, called CogTool-Explorer, which considers the layout position and the grouping of options in the user-interface in concert with a serial evaluation visual search process and information scent. Tests show that predictions from CogTool-Explorer match participant data better than alternative models that do not consider layout position and grouping. This dissertation work has also integrated the CogTool-Explorer model into an existing modeling tool, called CogTool, making it easier for other researchers and practitioners to setup and generate predictions of likely user exploration paths and task performance using CogTool-Explorer.

ACKNOWLEDGEMENTS

My advisor, **Bonnie John**. Thank you for the guidance, critiques, encouragements and opportunities that you have so generously given me, for the space and time I need to learn what must have been so obvious to you, and for keeping me on track towards the light at the end of the tunnel.

My thesis committee members (in alphabetical order), **Aniket Kittur**, **Brad Myers** and **Peter Pirolli**. Thank you for the insightful comments and questions that made my dissertation better, and for your patience and words of encouragement as I worked on my research.

Creator of AutoCWW, **Marilyn Blackmon**, and the creators of SNIF-ACT, **Wai-Tat Fu** and **Peter Pirolli**. Thank you for such inspirational work that seeded my research and is the standard I will continue to strive towards. Thank you for all the help and advice you have given me.

Members of the CogTool software development team (in alphabetical order), **Brett Harris**, **Michael Horowitz**, **Don Morrison** and **Ryan Myers**. Thank you for helping make CogTool-Explorer so much easier to use.

Esteemed researchers (in alphabetical order), **Rachel Bellamy**, **Stuart Card**, **Ed Chi**, **Wayne Gray**, **Tim Halverson**, **Anthony Hornof**, **Andrew Howes**, **Peter Polson**, **Dario Salvucci**, **Vladislav "Dan" Veksler**, **Richard Young**, and many others. From in-depth discussions to simple words of encouragement, thank you for making a difference in my PhD journey.

My mentor, **Naresh Kumar**. Thank you for the many opportunities that you have provided me to grow and mature professionally.

My parents, Mr. **Teo Thio Koon** and Mdm. **Lee Wai Chun**. Thank you for your countless sacrifices and selfless love, and for nurturing me in my formative years.

My wife, **Yeo Lay**. Thank you for sharing my pains and joys throughout these years as I worked on my PhD, for the *sayangs* when I needed comforting, and for the constant encouragement that kept me going. Thank you for being my best friend.

My spiritual teacher and friend, **His Holiness Yisie Vorle Kunsang Jigme Dorje**. Thank you for your teachings and showing me the Path.

TABLE OF CONTENTS

Abstract iv

Acknowledgements v

Table of Contents vi

List of Figures and Tables xi

1 Introduction 1

1.1 Research Gaps in Modeling 1

1.2 Modeling Tool for Practitioners 2

1.3 Organization of Dissertation 3

2 Related Work 4

2.1 Observational Studies and Surveys 4

2.2 Experimental Studies 5

2.3 Theories and Models 8

2.3.1 CE+, IDXL and LICAI+ 10

2.3.2 CoLiDeS 11

2.3.3 Information Foraging Theory, WUFIS and SNIF-ACT 12

2.3.4 MESA 14

2.3.5 Exploratory Act and Normalization Assumption 14

2.3.6 DOI-ACT 15

2.3.7 Research Progression in Prior Theories and Models 17

2.4 Visual Search 18

2.4.1 Eccentricity and Inhibition of Return 18

2.4.2 Grouping 19

2.4.3 Visual Search in User Exploration 20

2.5 Device Models 21

2.6 Modeling Tools 23

3 Research Gaps 25
3.1 Consideration of Layout Position 25
3.2 Consideration of Grouping 28
3.3 Implementation as a Tool 29
4 CogTool-Explorer 31
4.1 Consideration of Layout Position 32
4.1.1 Add Perceptual-Motor Behavior 32
4.1.2 Add a Visual Search Strategy 33
4.1.3 Preserve Layout Position 34
4.1.4 Operation of CogTool-Explorer 1.0 34
4.1.5 Test of CogTool-Explorer 1.0 34
4.2 Consideration of Grouping 38
4.2.1 Multi-Page Layout 41
4.2.1.1 Operation of CogTool-Explorer 1.0 in the Multi-Page Layout 42
4.2.1.2 Test of CogTool-Explorer 1.0 44
4.2.1.2.1 Task Performance Measures and Comparison Metrics 44
4.2.1.2.2 Comparison Results 46
4.2.1.3 Refinement of Infoscent Estimate for Top-Level Links 48
4.2.1.4 Addition of Reselection Behavior 52
4.2.1.5 Refinement of GoBack Behavior 56
4.2.1.5.1 Addition of Confidence 59
4.2.1.5.2 Updating of Confidence 63
4.2.1.6 Alignment to Human-scale Speed of Execution 66
4.2.1.6.1 Refinement of Perceptual-Motor Requirements in Task 66
4.2.1.6.2 Refinement of Duration to Access Information Scent 69
4.2.1.7 Summary of the Performance of CogTool-Explorer 1.1 72
4.2.2 Half-flatten Layout 75
4.2.2.1 Enable Group Relationships in CogTool-Explorer 1.2 77
4.2.2.2 Consider Groups: Operation of CogTool-Explorer 1.2 78

4.2.2.3 Ignore Groups: Operation of CogTool-Explorer 1.2 81
4.2.2.4 Test of CogTool-Explorer 1.2 82
4.2.3 Multi-Group Layout 85
4.2.3.1 Consider Groups: Operation of CogTool-Explorer 1.2 86
4.2.3.2 Ignore Groups: Operation of CogTool-Explorer 1.2 88
4.2.3.3 Test of CogTool-Explorer 1.2 89
4.2.4 Comparison of CogTool-Explorer to AutoCWW's Predictions 92
4.2.4.1 Multi-Page Layout 93
4.2.4.2 Half-flatten Layout 95
4.2.4.3 Multi-Group Layout 98
4.2.5 Summary of Results 100
4.3 Integration of CogTool-Explorer into CogTool 102
4.3.1 Import of Webpages from Websites 102
4.3.2 Specification of Groups 104
4.3.2.1 Textual Cues of Widgets, Groups and Remote Labels 105
4.3.3 Retrieval of Information Scent Scores 108
4.3.4 Setup of Model Runs 110
4.3.5 Presentation of Model Runs 112
4.3.6 Another Usage Example of the Modeling Tool 112
4.3.7 Further Design Iteration of Modeling Tool User-Interface 115
5 Contributions, Limitations and Future Work 116
5.1 Contributions 116
5.2 Verification at the Mouse-Click Level 116
5.3 Layout-specific Knowledge 117
5.4 Tasks, Parameters and Generality 118
5.5 Information Scent Scores 118
5.6 Other Factors 119
References 120

Appendices 125

A-1: CogTool Project Window 126

A-2: Predict Novice Exploration Dialog 127

A-3: Predict Novice Exploration Dialog: Task Tab 128

A-4: Predict Novice Exploration Dialog: Simulated User Tab 129

A-5: CogTool-Explorer Results and Settings 130

A-6: CogTool-Explorer Model Runs 131

A-7: CogTool-Explorer Collapsed View 132

A-8: CogTool Task Group 133

A-9: CogTool Task Group with CogTool-Explorer Task 134

A-10: Rename CogTool-Explorer Task 135

A-11: CogTool-Explorer Expanded View 136

A-12: Predict Novice Exploration from Existing KLM Model 137

A-13: Automated Creation of Task Group 138

A-14: Predict Novice Exploration from Existing CogTool-Explorer Task 139

A-15: CogTool-Explorer Results and Settings 140

A-16: Change Result Summary Format for CogTool-Explorer Task 141

A-17: Percent Success of CogTool-Explorer Task 142

A-18: CogTool-Explorer Expanded View and Comments Fields 143

B-1: Visualization of CogTool-Explorer Model Runs 144

B-2: Visualization of CogTool-Explorer Model Runs: Filtered #1 145

B-3: Visualization of CogTool-Explorer Model Runs: Filtered #2 146

B-4: Comparison of CogTool-Explorer Model Runs 147

B-5: Comparison of CogTool-Explorer Model Runs: Filtered #1 148

B-6: Comparison of CogTool-Explorer Model Runs: Filtered #2 149

C: Source Code of CogTool-Explorer 1.2 150

D: Python Script to Process CogTool-Explorer 1.2 Log Files 182

FIGURES

Figure 1: The Cricket Graph menu interface (from Rieman et al., 1996) 6

Figure 2: Example of a main webpage layout in Blackmon et al. (2002; 2003; 2005) 7

Figure 3: Example of a 2-level hierarchy layout in Blackmon et al. (2002; 2003; 2005) 8

Figure 4: DOI-Tree Visualization (Budiu & Pirolli, 2007) 16

Figure 5: An example layout with semantically cohesive groups, group labels and background color in Halverson and Hornof (2008) 20

Figure 6: Mockup of a website UI design in CogTool. CogTool can convert the mockup into an ACT-R device model that an ACT-R model can interact with. 22

Figure 7: A search goal and webpage used in AutoCWW Experiment 2 (Blackmon et al., 2002). Each search goal had one correct link on the webpage. The correct link for the goal "Canon law" is "Theology & Practices". AutoCWW identified links "Religious Figures" and "Religions & Religious Groups" as competing links that are also semantically related to the goal. The left picture shows the original layout. The right picture shows a modified layout with links "Theology & Practices" and "Religious Figures" swapped. The hypothesis is that users would be more likely to click on the correct link in the modified layout than in the original layout. 26

Figure 8: Participants' mean clicks on webpage and percent first click success by target column (Standard error shown) 27

Figure 9: Overview of CogTool-Explorer. White background indicates pre-existing components and external resources. Light blue indicates new or refined components contributed by this dissertation. 31

Figure 10: An example run of CogTool-Explorer 1.0 in the two-column layout. 35

Figure 11: Mean clicks on webpage by target column, comparing participants' performance to predictions by CogTool-Explorer 1.0 and AutoCWW (Standard Error shown) 36

Figure 12: Percent first click success by target column, comparing participants' performance to predictions by CogTool-Explorer 1.0 (Standard error shown) 37

Figure 13: An example of flattening two levels of webpages (left) into a single webpage (right) by grouping the links from each 2nd-level webpage onto a single webpage. These webpage examples are from AutoCWW experiments in Blackmon et al. (2005) and Toldy (2009). 38

Figure 14: (a) The multi-page layout where each link in the top-level webpage (also referred to as a top-level link) led to its corresponding 2nd-level webpage of links, (b) the half-flatten layout where selecting a top-level link reveals the 2nd-level links grouped under that top-level link, and (c) the multi-group layout where the 2nd-level links are grouped into nine groups. 39

Figure 15: The Microsoft Encarta Website 40

Figure 16: In the multi-page tasks, participants start on the top-level page (leftmost) and on selecting a link, transits to 2nd-level pages. Participants may go back to the top-level page, or may select a link to go to its 3rd-level page. In a 3rd-level page, participants can check if they have succeeded in the task, and if not, go back to the 2nd-level page and continue exploration. 41

Figure 17: An example run of CogTool-Explorer 1.0 in the multi-page layout. 42

Figure 18: CogTool-Explorer 1.0 compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 48

Figure 19: CogTool-Explorer 1.0a compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 51

Figure 20: CogTool-Explorer 1.0b compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 54

Figure 21: LinkClicks by CogTool-Explorer 1.0b compared to participant data, broken down by the four quartiles shown in Table 5. Each data point represents a link in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 55

Figure 22: CogTool-Explorer 1.0c compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 59

Figure 23: CogTool-Explorer 1.0d compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 62

Figure 24: CogTool-Explorer 1.0e compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 65

Figure 25: Part of the mock-up of the multi-page layout in CogTool-Explorer 1.0e. The entire mock-up is composed of 103 webpages (one top-level page, nine 2nd-level pages and ninety-three 3rd-level pages). This figure shows the top-level page, a 2nd-level page and a 3rd-level page. 67

Figure 26: Part of the mock-up of the multi-page layout in CogTool-Explorer 1.0f. The entire mock-up is composed of 103 webpages (one top-level page, nine 2nd-level pages and ninety-three 3rd-level pages). This figure shows the top-level page, a 2nd-level page and a 3rd-level page. 68

Figure 27: CogTool-Explorer 1.1 compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 70

Figure 28: Results over seven iterations from CogTool-Explorer 1.0 to CogTool-Explorer 1.1 in the multi-page layout. The blue diamond markers and the left vertical axis are for Correlation, where higher is better. The red square markers and the right vertical axis are for % Average Absolute Error, where lower is better. 74

Figure 29: In the half-flatten tasks, participants start in the top-level page (leftmost) and on selecting a link, transits to 2nd-level pages. Participants can see both top-level links and 2nd-level links in 2nd-level pages. Participants may select a top-level link to go to another 2nd-level page, or may select a 2nd-level link to go to its 3rd-level page (not shown here). In a 3rd-level page, participants can check if they have succeeded in the task, and if not, go back to the 2nd-level page and continue exploration. 75

Figure 30: Blue rectangles show the visual elements available at different stages of a group-based hierarchical exploration process in the half-flatten layout. Figure (a) shows on transiting to a 2nd-level page and Figure (b) shows after going back from the group of 2nd-level links. 76

Figure 31: An example run of Consider Groups by CogTool-Explorer 1.2 in the half-flatten layout. 79

Figure 32: An example run of Ignore Groups by CogTool-Explorer 1.2 in the half-flatten layout. 82

Figure 33: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the half-flatten layout. Each data point in (a) (b) (c) and (d) represents a task, and in (e) and (f) a link in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 84

Figure 34: In the multi-group tasks, participants can see all nine groups of links on the top-level page. Selecting a link will transit to 3rd-level pages like those in the multi-page and half-flatten layouts (not shown here). In a 3rd-level page, participants could check if they had succeeded in the task, and if not, go back to the top-level webpage and continue exploration. 85

Figure 35: An example run of Consider Groups by CogTool-Explorer 1.2 in the multi-group layout. 87

Figure 36: An example run of Ignore Groups by CogTool-Explorer 1.2 in the multi-group layout. 89

Figure 37: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the multi-group layout. Each data point in (a) (b) (c) and (d) represents a task, and in (e) and (f) a link in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 91

Figure 38: Results of *MeanClicksToSuccess* by CogTool-Explorer 1.1 and AutoCWW, compared to participant data in the multi-page layout. Each data point represents a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 94

Figure 39: Results of *MeanClicksToSuccess* by CogTool-Explorer 1.2 and AutoCWW, compared to participant data in the half-flatten layout. Each data point represents a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 97

Figure 40: Results of *MeanClicksToSuccess* by CogTool-Explorer 1.2 and AutoCWW, compared to participant data in the multi-group layout. Each data point represents a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points. 99

Figure 41: In CogTool's Frame Window, users can drag and drop standard UI widgets such as buttons and links from a palette of widgets onto a frame. A frame represents a display state in the UI. In CogTool's Design Window, users can create different frames and specify how interface actions on a widget, such as a mouse click, changes the display state of the UI, by drawing transitions from that widget in its frame to another frame. 103

Figure 42: Import HTML Pages dialog box 104

Figure 43: To specify existing widgets form a group, the practitioner would multiple select these widgets and then issue the "Group" command from either the "Modify" menu or its keyboard shortcut. This will create a group composed of the previously selected widgets and the newly created group will be automatically selected with the group's bounding box highlighted in green (Figure a). Selecting a widget that is a member of a group will highlight the widget with a gray border (which is standard CogTool behavior) and also highlight the group with a red border (Figure b). 105

Figure 44: The multi-group layout with the Performing Arts group selected, highlighted by a green bounding box. 107

Figure 45: The multi-group layout with the Remote Label "Performing Arts" selected, highlighted by a gray bounding box. Selecting a Remote Label will also highlight its owner, in this example the Performing Arts group, with a red bounding box. 108

Figure 46: Select "Generate Dictionary" from the cell at the intersection of a Task Description and a UI mockup to generate infoscent scores for every Displayed Label and textual-cue in the UI design. The modeler can inspect and modify parameters and infoscent scores using the Dictionary Viewer window (bottom of Figure). For example, the highlighted row in the Viewer has its cosine parameter changed from 0.5 to 1.5 to disable further elaboration. 109

Figure 47: Select "Recompute Script" from the cell at the intersection of a Task Description and a UI mockup to bring up the dialog box to set up model runs, including the number of runs to do, parameter values such as *k* ("Eagerness"), the frame to start exploration from, and the frame or frames that indicate a successful model run. Each model run will be listed in CogTool's Project Window and can be further inspected in CogTool's Script Step Window. 111

Figure 48: Mockup of the Control Display Unit (CDU) of a Boeing 777's cockpit. An image of the CDU was used as the background image of this frame. Button widgets were created over the parts of the image where a real button would be and text label widgets were created over the parts of the image where textual information would be displayed in the CDU's display. Note that the text string "Initialization Reference position", which describes the Displayed Label "INIT REF", was entered as part of the Displayed label before the Auxiliary Text field was implemented in CogTool-Explorer. Here, the "INIT REF" button is selected (highlighted with a gray border, which is standard CogTool behavior) and the Mode Keys group that the "INIT REF" button belongs to is highlighted with a red border (which is new in CogTool-Explorer). 113

TABLES

Table 1: Comparison of related work and CogTool-Explorer by scope of exploratory behavior, modeling approach, factors considered, extent of model implementation and implementation as a modeling tool for practitioner use. A white cell indicates non-consideration and darker shades indicate different or progressively more sophisticated consideration. See Notes for details. 9

Table 2: CogTool-Explorer 1.0 compared to participant data in the multi-page layout 46

Table 3: CogTool-Explorer 1.0 and 1.0a compared to participant data in the multi-page layout. The better results are highlighted in bold. 50

Table 4: CogTool-Explorer 1.0a and 1.0b compared to participant data in the multi-page layout. The better results are highlighted in bold. 53

Table 5: The 36 tasks in the multi-page layout sorted by decreasing %Success by participants and grouped into 4 quartiles of 9 tasks each. Table also shows the breakdown of %Success and LinkClicks by CogTool-Explorer 1.0b in these 4 quartiles. 55

Table 6: Percentage of go-back actions from 2nd-level pages by CogTool-Explorer 1.0b as a function of the GoBackCost parameter, compared to participants. CogTool-Explorer 1.0c uses the GoBackCost parameter value of 1. 58

Table 7: CogTool-Explorer 1.0b and 1.0c compared to participant data in the multi-page layout. The better results are highlighted in bold. 58

Table 8: CogTool-Explorer 1.0c and 1.0d compared to participant data in the multi-page layout. The better results are highlighted in bold. 62

Table 9: CogTool-Explorer 1.0d and 1.0e compared to participant data in the multi-page layout. The better results are highlighted in bold. 65

Table 10: CogTool-Explorer 1.0e and 1.1 compared to participant data in the multi-page layout. The better results are highlighted in bold. 70

Table 11: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the half-flatten layout. The better results are highlighted in bold. 83

Table 12: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the multi-group layout. The better results are highlighted in bold. 90

Table 13: CogTool-Explorer 1.0 and 1.1 compared to participant data in the multi-page layout. The better results are highlighted in bold. 100

Table 14: CogTool-Explorer 1.1 compared to participant data in the multi-page layout, and CogTool-Explorer 1.2 compared to participant data in the half-flatten and multi-group layouts. The best results are highlighted in bold. 101

Table 15: Seven iterations of a model, the changes made in CogTool-Explorer to produce each iteration, the time it took to make those changes, and the %Success of the resulting model. (Adapted from Table 1, John et. al., 2009) 114

1 INTRODUCTION

Designing user-interfaces (UIs) so that first-time or infrequent users can accomplish their goals by exploration has been an enduring challenge in Human-Computer Interaction (HCI). Early examples include automated teller machines and public information kiosks that people walk up and use. A more contemporary example is the World-Wide-Web (WWW or Web), where users may be skilled in general website navigation, but lack prior knowledge of where the desired information resides in varied and constantly evolving websites. Both Franzke (1995) and Cox (2002) suggested that users, who have some experience in a software application or device, or other similar software applications or devices, may be more likely to attempt and be successful at exploration. These users may only use the software sporadically or are only familiar with parts of the software, compared to skilled users who know exactly what to do in the UI to accomplish a task, and novice users who lack too much knowledge in both the design of the UI and the task domain to be successful in exploration. To support this group of users who are not complete experts or novices, UI design should support use through exploration.

Fortunately, HCI practice and research has risen to this challenge. Iterative user-testing is an effective but costly method to develop UIs that support use through exploration. A complementary method is to use modeling tools that can generate predictions of user exploration given a UI and a task. Predictive modeling tools can help evaluate and weed out a larger number of early UI designs. Model predictions can also inform the design process by highlighting the probable causes that might lead to poor user task performance. Predictive human performance models have been successfully used to evaluate and design UIs for skilled routine interactive tasks (John & Kieras, 1996). If users have to explore and learn an unfamiliar UI, models such as CE+ (Polson and Lewis, 1990), IDXL (Rieman, Young & Howes, 1996) and LICAI+ (Kitajima, Soto & Polson, 1998) describe how knowledge representation, mental model construction and label-following drive user behavior during exploration. More recent models like MESA (Miller & Remington, 2004) and SNIF-ACT 2.0 (Fu & Pirolli, 2007), and tools like Bloodhound (Chi et al., 2003) and AutoCWW (Blackmon, Kitajima & Polson, 2005) have focused on predicting user exploration of websites and demonstrated how predictions can inform UI design. These models of user exploration employ the common concepts of *label-following* and *information scent*, that the user's choice is partly determined by the semantic relevance between the user's goal and the options presented in the UI.

However, there are other factors beside information scent (or *infoscent* for short) that influence exploration and the likelihood of success, and models and tools that do not consider these factors would make less accurate predictions. The goal of this dissertation is to include some of these other factors in a process model and modeling tool for more accurate predictions of user exploration, to better inform UI design.

1.1 RESEARCH GAPS IN MODELING

In addition to infoscent, the layout of the UI also affects the choices made during exploration, because a user is not likely to select an option that he or she did not look at and evaluate, and competing options (i.e. options that are semantically relevant to a particular task but are incorrect for the task) seen before the correct option might be chosen instead. Furthermore, the spatial and

semantic grouping of options in a UI also affects user exploration, where competing options nested inside a group with a competing group heading (i.e. a text label identifying a group) was found to be especially problematic for successful exploration (Blackmon et al., 2005; Kitajima, Polson & Blackmon, 2007). Although infoscent, visual search, layout position and grouping affect user exploration, prior research like Bloodhound and AutoCWW consider only a subset of these factors in their model predictions, which could result in inaccurate predictions when these factors matter and interact in the UI and task. For example, Section 3.1 describes an analysis of participant data and AutoCWW predictions that was done as part of this dissertation, where predictions did not match data at a more detailed level of analysis when visual search and layout position were not considered together with infoscent.

To address these gaps, this dissertation contributes a new modeling tool called *CogTool-Explorer*, to make more accurate predictions of goal-directed user exploration compared to prior modeling tools. The research work in this dissertation primarily focuses on the modeling and predictions by CogTool-Explorer, with a secondary focus on making CogTool-Explorer into a tool for practitioner use. The thesis of this dissertation is:

> *A modeling tool for goal-directed user exploration of user-interfaces, that considers the information scent, the visual search process, and the layout and grouping of options in the user-interface in concert, can make more accurate predictions of user exploration compared to tools that do not.*

1.2 Modeling Tool for Practitioners

To encourage HCI practitioners to use modeling tools alongside user testing and other HCI methods, it must be easy to set up the model for a given UI and task, run the model and obtain prediction results. However in most prior research, such as CE+, IDXL and LICAI+, the research effort focused on implementing, testing and refining theory, and was not explicitly concerned with the implementation of a more flexible tool for modeling UI designs beyond the designs used in the particular research project. Later research on tools like Bloodhound and AutoCWW made it possible for practitioners to use Bloodhound's and AutoCWW's underlying predictive models without the need to write or edit software code. Both tools are tailored to model exploration of webpages and websites. In Bloodhound, the practitioner can specify an entire website by entering its Web address. In AutoCWW[1], the practitioner has to manually enter the text label of links from the webpages being modeled.

The approach in this dissertation is to implement CogTool-Explorer as part of an existing modeling tool called CogTool (John, Prevas, Salvucci & Koedinger, 2004). CogTool is a publicly available modeling tool[2] that can predict skilled performance times from tasks demonstrated on a mockup of a UI design. Compared to earlier established modeling methods, CogTool reduced the time taken by both expert and novice modelers to create a correct model of skilled interactive behavior by an

[1] Accessed on January 17, 2011 at http://autocww.colorado.edu/~brownr/ACWW.php

[2] Downloaded on January 17, 2011 from http://cogtool.hcii.cs.cmu.edu/

order of magnitude, and significantly reduced the variation between novice modelers in predicting the task execution time of skilled interactive behavior (John, 2010). In CogTool, the practitioner can create a mockup of a UI by dragging and dropping standard UI widgets such as buttons, menus and links, from a palette of widgets onto frames. A frame represents a display state in the UI. The practitioner can specify how interface actions on a widget, such as a mouse button press or click, changes the display state by drawing transitions from that widget in its frame to another frame. The practitioner can easily create a predictive model of skilled behavior by demonstrating the exact interaction steps of a task in the UI mockup. On a single command to "go", CogTool will run the model and generate the prediction.

In CogTool-Explorer, the practitioner will provide a text description of the exploration goal and on a single command to "go", the CogTool-Explorer model will explore and interact with the UI mockup to attempt to complete the task. The practitioner can then view and save the prediction results from CogTool-Explorer. To support the new requirements and functionality of CogTool-Explorer, this dissertation adds to CogTool new menu options, dialogs and supporting modules for creating large UI mockups, specifying group relationships in the UI mockup and to generate the infoscent scores that will be used by the model. Through this integration of CogTool-Explorer into CogTool, the hope is that it will extend the success of CogTool, and thus CogTool-Explorer, in delivering usable modeling to HCI practitioners.

1.3 Organization of Dissertation

Section 2 reviews related work on goal-directed user exploration in HCI, focusing on their key findings and limitations. Section 3 synthesizes the outcomes from the review of related work and highlights the three research gaps that are the focus of this dissertation. Section 4 presents CogTool-Explorer, describing in detail the modeling and implementation work done to address these research gaps, and the test results from comparing CogTool-Explorer to both human data and AutoCWW predictions. Section 5 concludes with the contributions, limitations and future work of this dissertation research. Appendices A and B present further UI designs that pertain to the integration of CogTool-Explorer into CogTool, and Appendices C and D contain the source code of the CogTool-Explorer model and the script that was used to process the model's log file.

2 Related Work

Section 2.1 reviews related observational studies and surveys that found evidence for goal-directed user exploration in various real-world computing task environments. Section 2.2 reviews related experimental studies that identified some of the factors that influence the choices made during user exploration. Section 2.3 reviews theories and models that were developed to explain the user exploration process and replicate results from the experimental studies. Section 2.4 reviews findings from eye-tracking studies of visual search that have a bearing on user exploration. Section 2.5 reviews related work to develop accurate device models of the UI on which exploration by a model takes place. Section 2.6 reviews related work to develop modeling tools intended for use by HCI practitioners.

2.1 Observational Studies and Surveys

Rieman (1994; 1996) had 14 participants keep a week-long log of their daily work activities, to find out how often they had to explore in their normal work activities. Whenever the participants "learned something new about their computer system or some other equipment in their work" (Rieman, 1994, p. 51), they were to record the event and the strategy or strategies they used. A surprising finding was the low occurrence of about 1 learning event for every 8 hours of computing time, although Rieman noted that the recorded events were of varying complexities and at different grain sizes, so a single recorded event could have been a number of individual events. The interesting finding was "the similarity in the learning strategies recorded across a very wide range of situations and users" (Rieman, 1994, p. 59) and that there was "significant evidence that the three preferred strategies are *trying things out, reading the manual* and *asking for help*" (Rieman, 1994, p. 59). Participants reported using the strategy of "tried different things until it worked" for over half (37 out of 60) of all recorded learning events in computer-based activities.

At the end of the dairy study, Rieman conducted structured interviews with the participants. When asked "when you get a new piece of software, how do you learn to use it?" (Rieman, 1994, p. 62), half the participants (7 out of 14) identified "exploring its functionality, usually in the context of actual tasks" (Rieman, 1994, p. 63) as one of the ways to familiarize themselves with the software. When asked how they figured out the way to do something they did not know in a program they already knew, more than half of the participants (9 out of 14) identified *trying things out* or *exploration* as the first strategy they used. The interviews also revealed that when the participants used the exploration strategy, all but one (13 out of 14) did so in the context of a task. Participants felt that goal-directed exploration was more productive and could be made relevant to actual work activities. Only one participant reported doing task-free exploration of new software.

The above investigations by Rieman were before the Web became a major real-world computing activity. Byrne, John, Wehrle and Crow (1999) conducted a video and verbal protocol study of Web use and found that the times spent on *Locate* (finding that information or link on a webpage, which typically requires some visual search) and *Go To* (any activity which caused the browser to display a particular webpage) activities were ranked second and third highest, after the time spent on *Use information* activities. In contrast, the time spent on *Configure* (changing the state of the browser, such as the size, location and number of browser windows) and *React* (when the browser demands

something of the user typically in the form of responding to a dialog box) activities combined was less than any one of these top three activities. This suggests that exploration on the Web is more about finding and using information. The focus of exploration is now on the hyperlinked UI of varied and constantly evolving websites and not on the UI of the browser application.

Morrison, Pirolli and Card (2001) analyzed a survey on Web activities and found that 96% of information seeking activity on the Web was searching for a particular or multiple pieces of information triggered by a goal. The other 4% were repeated visits to monitor information updates and general searching for information not triggered by a goal. This suggests that information seeking on the Web, like the exploration of software and devices by the participants in Rieman (1994; 1996), was almost always done in the context of a goal-directed task.

The results from these studies suggest that goal-directed user exploration does indeed happen in real-world computing tasks. Exploration of the UI to learn a new piece of software or to figure out how to perform a task in existing software was a preferred approach; a strategy at least as often used as reading manuals and asking for help. In user activities on the Web, exploration of websites to locate desired information was the dominant activity, which is not surprising considering the varied and constantly evolving designs of websites. In both cases, exploration was almost always done in the context of a task goal.

2.2 Experimental Studies

Franzke (1994; 1995) conducted one of the first detailed experiments on computer users using unfamiliar software for the first time. Seventy-six participants, who all had experience with Macintosh computers but had never used a graphing software application, were given a task description to draw and modify graphs using one of four graphing software applications. Two of these were versions of an application called Cricket Graph (Figure 1), which was the subject of modeling in later research (discussed in the review of IDXL and LICAI+ in Section 2.3.1). The key result from the study was that participants took more time to select the correct option on the screen if the *semantic distance* between the label of the correct option and the task description was larger. Franzke defined 4 levels of semantic distance in increasing order:

1. Overlap – words that were identical to words as presented in the task description, such as if the goal was to "create a graph" and the menu item had the label "graph"

2. Synonym – words that were synonymous to words in the task description, such as if the goal was to "create a graph" and the menu item had the label "chart"

3. Inference – words that were semantically related to words in the task description but required some inference, such as if the goal was to "create a graph" and the menu item had the label "drawing tools"

4. No Link – words that had no direct semantic link to words in the task description, such as if the goal was to "create a graph" and the menu item had the label "file".

Franzke also identified two other factors that increased the time participants took to select the correct option on the screen. One was the number of options on the screen. The more options there

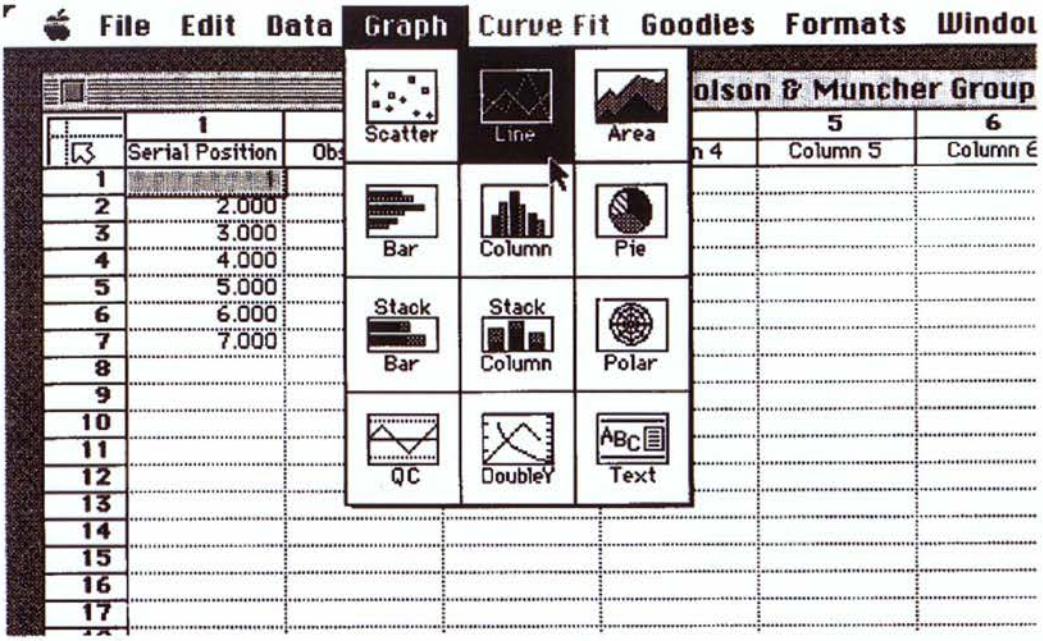


Figure 1: The Cricket Graph menu interface (from Rieman et al., 1996)

were, the longer the exploration time, especially if the correct option had a poor label (i.e. large semantic distance from the goal). The other factor was hidden options, where participants "took longer to discover direct manipulation interactions on unlabeled objects, such as double-clicks on graph objects, drag-and-drop operations, etc." (Franzke, 1995)

Blackmon and colleagues (2002; 2003; 2005) and Kitajima et al. (2007) conducted and analyzed a series of experiments where participants were presented with the text description of a search goal for an encyclopedia article and asked to navigate a main webpage (Figure 2) or a 2-level hierarchy of webpages (Figure 3) to find the target webpage which contained the article. The link labels were short texts describing encyclopedia topics. The topics links were further grouped into related topics and each group was given a non-interactive heading label (on the main webpage) or an interactive heading link (in the 2-level hierarchy of webpages). Selecting a topic link would display a webpage that listed all the encyclopedia articles under that topic.

Blackmon et al. used Latent Semantic Analysis (LSA; Landauer, McNamara, Dennis & Kintsch, 2007) to compute an engineering approximation (LSA cosine value between two LSA document vectors) of the semantic similarity between the text of each topic link and the goal description, and between the text of each heading label/link and the goal description. They found that participants took more clicks to find the target webpage when the target webpage was located under a topic link that was computed to be weakly related to the goal, and took less clicks when the topic link was strongly related to the goal. This result is in agreement with Franzke's semantic distance, except that Franzke scored the semantic distances between on-screen labels and the task description by hand. Blackmon et al. also used LSA to compute an engineering approximation of participants' familiarity with the text of each topic link (LSA term vector length and term frequency count in the semantic space that represented the reading knowledge level of the participants). They found that

Find encyclopedia article about Audiometer

Audiometer, instrument for testing hearing. The audiometer is an essentially simple instrument that produces pure tones of various fixed pitches (frequencies) heard through headphones. Hearing is tested one ear at a time. The operator can switch between frequencies and repeat the process with each frequency. Typically, sensitivity may be tested at frequencies of 125 hertz (Hz, or cycles per second), 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz, 8000 Hz, and 12,000 Hz. As an alternative to testing the normal mode of hearing through headphones, hearing by bone conduction can be tested. Hearing is never uniform over all frequencies and commonly varies widely at different frequencies. Internally, audiometers consist of a transistorized, variable-frequency audio oscillator—usually a simple feedback device—capable of producing a sinusoidal (near sine-wave) output.

Sports, Hobbies, & Pets	Performing Arts	Religion & Philosophy
Sports Sports Figures Games, Hobbies, & Recreation Pets	Theater Musicians & Composers Cinema, Television, & Broadcasting Music Dance Musical Instruments	Theology & Practices Mythology Religious Figures Philosophy Religions & Religious Groups Scripture The Occult
Art, Language & Literature	**Geography**	**History**
National & Regional Literature Literature & Writing Architecture Artists Language Writers & Poets Decorative Arts Legends & Folklore National & Regional Art Painting, Drawing, & Graphic Arts Sculpture Periods & Styles Photography	World Cities, Towns, & Villages Regions of the World Rivers, Lakes, & Waterways Parks & Monuments Countries Canadian Provinces & Cities Islands Mountain Ranges, Peaks, & Landforms U.S. Cities, Towns, & Villages Maps & Mapmaking Oceans & Seas Exploration & Explorers U.S. States, Territories, & Regions	History of Asia & Australasia People in European History People in United States History United States History African History World History & Concepts Ancient History History of the Americas European History
Physical Science & Technology	**Life Science**	**Social Science**
Construction & Engineering Chemistry Earth Science Computer Science & Electronics Machines & Tools People in Physical Science Astronomy & Space Science Paleontology Industry, Mining, & Fuels Physics Transportation Communications Mathematics Military Technology Time, Weights, & Measures	Plants People in Life Science Medicine Invertebrate Animals Fish Algae & Fungi Agriculture, Foodstuffs, & Livestock Mammals Reptiles & Amphibians Biological Principles & Concepts Anatomy & Physiology Environment Birds Viruses, Monerans, & Protists	Economics & Business Organizations Institutions Political Science Psychology Law Education Anthropology Military Sociology & Social Reform Calendar, Holidays, & Festivals Archaeology

Figure 2: Example of a main webpage layout in Blackmon et al. (2002; 2003; 2005)

participants took more clicks if the correct topic link was computed to be unfamiliar to the participants.

Blackmon et al. (2005) further found that participants took more clicks to find the target webpage if there were *competing links* on the webpage (i.e. topic links that were computed to be similar to the goal but did not lead to the target webpage), especially when the competing links were nested under a *competing heading* (i.e. a heading label/link that was computed to be similar to the goal but whose group did not contain any topic links that led to the target webpage).

These experimental studies increased our understanding of one of the key factors that influence goal-directed user exploration, namely the semantic relevance of the options presented in the UI with respect to the exploration goal. For most well-designed UIs, a reasonable assumption is that options are appropriately labeled. By *label-following*, users are more likely to be successful in exploration if they select options with labels that are semantically relevant to the goal. Furthermore, while the semantics of individual options matter, the semantics arising from organizing structures such as groups and group headings also affect exploration success.

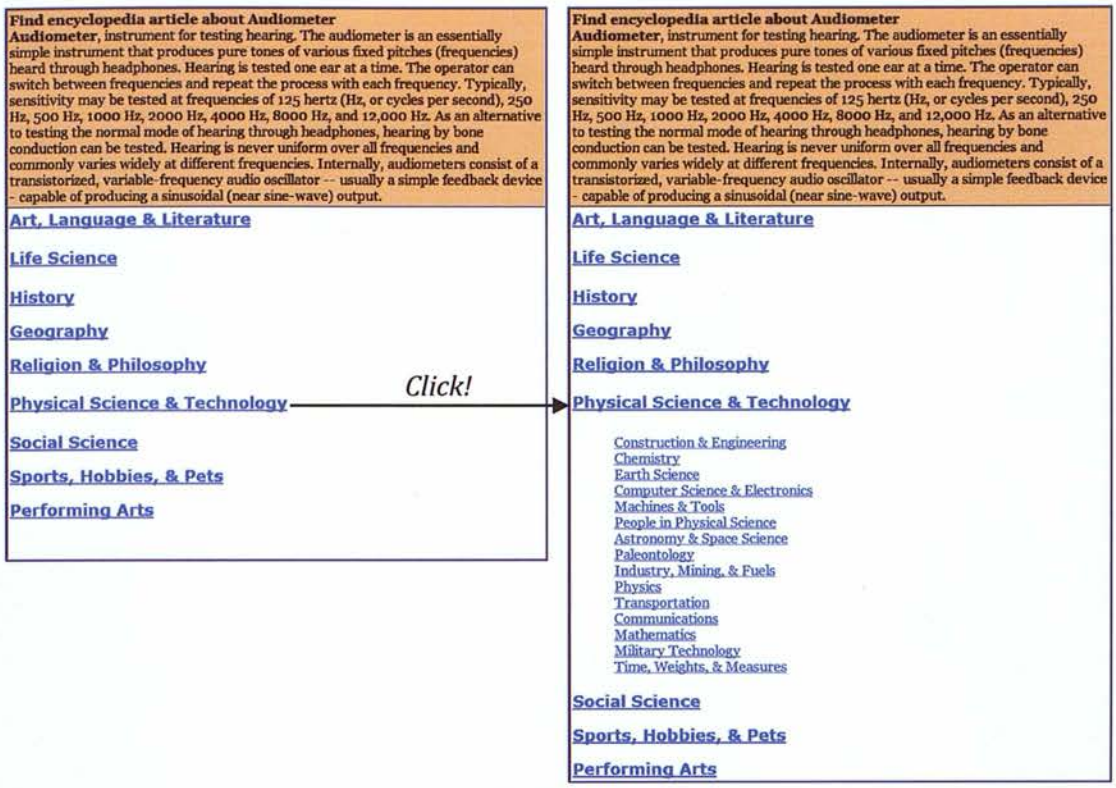


Figure 3: Example of a 2-level hierarchy layout in Blackmon et al. (2002; 2003; 2005)

2.3 Theories and Models

Theories and computational models of goal-directed user exploration seek to explain the findings and replicate the results from observational and experimental studies. Pioneering theoretical and modeling research was done in the larger context of skill acquisition in HCI through *learning by exploration*. The research question in CE+, IDXL and LICAI+ was how users explore and learn a new software application, and how learning by exploration affects further exploration and improves task performance.

The advent of the Web presents a different task environment for goal-directed exploration, where finding information from within webpages in websites is the dominant activity (Byrne et al., 1999; Morrison et al., 2001). Research in this domain focuses more on the intricacies of label-following in determining exploration success, and less on the discovery and learning of interface actions on UI widgets, possibly because the majority of webpages on the Web do not support interface actions other than a mouse click on a text or graphic link. Some models have a visual search process, and models differ in their modeling approach, derivation of infoscent, representation of the device UI and the extent of implementation. Table 1 provides a summary of the theories and models reviewed in the following sub-sections.

	Scope of Exploratory Behavior					Modeling Approach	Factors Considered			Model Implementation	Tool for Practitioner
	Goal Formation	Learning	Interface Action	Label-Following	Visual Search		Information Scent	Layout Position	Grouping		
						Global Evaluation and Decision	Human raters	Relative position		Contains non-executable components	
						Serial Evaluation and Local Decision	Mixed Approach	Mixed Approach		Executable components but not fully integrated	
							Algorithmically assigned	Absolute position		Executable Model	
Related Work (Year)											
CE+ (1990)											
IDXL (1996)											
LICAI+ (1998)											
CoLiDeS (2000)	[a]	[a]	[a]	[a]	[b]				[b]	[b]	
WUFIS (2001)											
Bloodhound (2003)											[c]
SNIF-ACT 1.0 (2003)											
MESA (2004)						[d]					
Cox's Model (2004)											
Bumbry's Model (2004)											
AutoCWW (2005)						[e]			[e]		
SNIF-ACT 2.0 (2007)					[f]			[f]			
DOI-ACT (2007)					[g]		[h]	[i]			
CogTool-Explorer											

Table 1: Comparison of related work and CogTool-Explorer by scope of exploratory behavior, modeling approach, factors considered, extent of model implementation and implementation as a modeling tool for practitioner use. A white cell indicates non-consideration and darker shades indicate different or progressively more sophisticated consideration. See Notes for details.

Notes:

[a] CoLiDeS inherits from LICAI+ and adds a visual attention cycle [b].

[b] CoLiDeS makes reference to visuospatial bottom-up processes and knowledge-driven top-down processes for its visual attention cycle and globally selects the optimal group or option.

[c] Bloodhound is not publicly available but is available for licensing.

[d] MESA is neutral as to the actual order in which the links are evaluated by randomly ordering links for each run and then taking performance averages across many runs.

[e] ACWW inherits its modeling approach from CoLiDeS and globally analyses all groups and links on a webpage for navigation problems.

[f] SNIF-ACT 2.0 assumes a left-right then top-down visual search path over the links on a webpage and encodes the ordering before the model runs.

[g] DOI-ACT prefers to shift visual attention further afield to the right, which assumes knowledge specific to the DOI tree UI layout, is different from the eccentricity heuristic favored by other models of visual search and may not be appropriate for other UI layouts in general.

[h] DOI-ACT uses information scent scores from human raters as well as algorithmically assigned scores.

[i] DOI-ACT assumes top-down visual search over the nodes in a group, but uses the absolute *xy*-coordinates of groups when evaluating and selecting a group.

2.3.1 CE+, IDXL AND LICAI+

Polson and Lewis (1990) combined three components into the CE+ model to give an account of learning by exploration:

1. Cognitive Complexity Theory (CCT; Kieras & Polson, 1985) to execute production rules and run the model. A production rule specifies a list of clauses and a list of actions. A rule will "fire" and execute its actions if the state of the model satisfies its clauses. Productions rules coordinate the execution of the second component.

2. A label-following variant of the *hill-climbing* strategy to choose among options. In hill-climbing, the strategy is to select the option that appears to offer the greatest progress toward the goal (Greeno & Simon, 1988). CE+ chooses the option whose description overlaps most with the goal, provided that the option has not been tried before.

3. EXPL (Lewis, 1986; 1988) to compare the prior display state of the system with the display state after an interface action was taken, and apply one of three heuristics to infer a causal relationship. This learned relationship is then encoded as a production rule which the CCT component may fire in the future.

While each of the above components was implemented as executable models in their own prior research, CE+ was not integrated into a single executable model (Table 1: Model Implementation). Instead, Polson and Lewis presented an account of a few hill-climbing and learning steps under the control of production rules in a "hand simulation" of the model.

Rieman et al., (1996) used both empirical observations from the Cricket Graph task in Franzke's (1994; 1996) experiment and theoretical arguments to define the Iteratively Deepening Exploratory Learning (IDXL) model. Like CE+, IDXL draws upon several separate models to account for the range of behavior in exploratory learning (Table 1: Scope of Exploratory Behavior). In the Cricket Graph task, participants were given a task description, but not step-by-step instructions, to draw and modify graphs using the software Cricket Graph. IDXL has an instruction-taking model of how participants might learn task instructions, and an analogy model of how participants can map instructions or past experience to novel situations.

IDXL has a guided depth-first search with iterative deepening (gDFID) strategy to model the exploration of pull-down menus in Cricket Graph that preceded the selection of a menu item (see Figure 1 for pull-down menus in Cricket Graph). Guided by the heuristic to limit its search to items semantically related to the task, the model serially scans the hierarchy of pull-down menus in an iterative-deepening process. The process iteratively scans deeper submenus and iteratively applies more costly comprehension methods to further evaluate menu items. At each step after a menu item is evaluated, the model may scan another menu item, further evaluate a menu item or choose a menu item that it found satisfactory. Rieman et al. stressed that large scale patterns of behavior should emerge out of local decisions. Through this serial evaluation process, IDXL's behavior reflects local decisions on individual labels, in contrast to LICAI/LICAI+'s approach (discussed below) of evaluating a large set of labels simultaneously (see Table 1: Modeling Approach).

In evaluating the IDXL model, Rieman et al. stated that the goal of IDXL was to achieve a qualitative match to behavioral patterns observed in participants. The implemented IDXL model addressed the first few steps of the Cricket Graph task, with the rest of the evaluation done by a qualitative comparison of the model's behavior to observed participant behavior (Table 1: Model Implementation).

Kitajima and Polson (1997) created the Linked Model of Comprehension-Based Action Planning and Instruction Taking (LICAI), the predecessor of LICAI+, to model the formation of an exploration goal from task instructions and the selection of actions during exploration in the same Cricket Graph task. LICAI is based on the Construction-Integration (CI; Kintsch, 1988; 1998) framework that is developed to model text comprehension. The basic mechanism is the CI cycle: the *construction* of a network of propositions representing the reader's goal, the next sentence read and existing knowledge, followed by the *integration* of this network by spreading activation through the network. The most activated nodes in the proposition network at the end of the integration represent the interpretation of the text.

LICAI uses the basic CI framework to model the formation of an exploration goal from task instructions as text comprehension. LICAI extends the CI framework to handle interface actions by encoding not just the labels of on-screen options in the proposition network, but also the actions afforded by these on-screen options. A new integration process first identifies the three most activated options, followed by a second integration to identify the most activated interface action. The proposition network contains all on-screen options at once. This means that LICAI globally identifies the three most optimal on-screen options, in contrast to the serial evaluation and localized decisions in IDXL (Table 1: Modeling Approach). The follow-up model, LICAI+ (Kitajima et al., 1998), adds the learning and recall of previous interface actions to affect future exploration.

CE+, IDXL and LICAI+ seek to provide an account for a range of user behaviors during exploration (Table 1: Scope of Exploratory Behavior). All three models are concerned with the discovery, learning and knowledge of interface actions. In IDXL and LICAI+, this might had been motivated by participants who were unaccustomed to the direct-manipulation intensive UI of a graphing software application such as Cricket Graph. All three models make exploration choices based on label-following. In these models, label attractiveness for the task was manually specified, whereas many subsequent models use automated methods to compute an engineering approximation of label attractiveness (Table 1: Information Scent). Subsequent models also use either the serial evaluation and localized decision approach like in IDXL, or the global evaluation and decision approach like in LICAI+ (Table 1: Modeling Approach).

2.3.2 CoLiDeS

Comprehension-based Linked-model of Deliberate Search (CoLiDeS; Kitajima, Blackmon & Polson, 2000; 2005) is an extension of LICAI/LICAI+ for Web navigation. CoLiDeS adds an attention CI cycle when a user encounters a webpage with many links (Table 1: Scope of Exploratory Behavior, notes [a] and [b]). The attention cycle parses the webpage into sub-regions and focuses attention on a sub-region most similar to the user's goal. The action selection CI cycle from LICAI then identifies the few most attractive links from the attended-to sub-region and then the most attractive interface

action. Both cycles globally identify the optimal choice. CoLiDeS also proposes five independent factors that combine to measure the attractiveness of a sub-region or a link with respect to the exploration goal:

1. The degree of semantic similarity between the user's goal and the sub-region's heading or link label. A more similar heading or link is more likely to be selected.

2. Whether there is an adequate level of relevant background knowledge to successfully elaborate the sub-region's heading or link label. A heading or link that triggers an inadequate level of relevant background knowledge is not likely to be selected.

3. Whether a word used in the heading or link label is a low-frequency term in the user's background knowledge. A heading or link that is a low-frequency term is not likely to be selected.

4. The frequency with which the user has encountered the screen widget or specific heading or link. Screen elements on frequently navigated paths are more likely to be selected.

5. Whether there is a literal matching, partial or complete, between the user's goal and the heading or link (e.g., looking for information about Type 2 Diabetes and seeing a link labeled "Type 2 Diabetes"). A heading or link with a literal match to the user's goal is more likely to be selected.

CoLiDeS has not been implemented into an executable model (Table 1: Model Implementation, note [b]), but has led to the development of an analytical method called Cognitive Walkthrough for the Web (CWW; Blackmon, Polson, Kitajima & Lewis, 2002) and a tool called Automated Cognitive Walkthrough for the Web (AutoCWW; Blackmon, Kitajima & Polson, 2005). AutoCWW implements the CoLiDeS concept of sub-regions, and the first three of the five factors listed above using LSA computed measures. However, information about sub-regions and links within sub-regions entered into AutoCWW by the practitioner does not indicate layout position (Table 1: Layout Position). Section 3.1 describes an analysis of participant data and AutoCWW predictions where predictions did not match data at a more detailed level of analysis because layout position was not considered. Section 2.6 describes the AutoCWW tool in more detail.

2.3.3 Information Foraging Theory, WUFIS and SNIF-ACT

Information Foraging Theory (Pirolli & Card, 1997; Pirolli, 2007) seeks to explain human information-seeking and information-usage behaviors, on the basis that information seekers are adaptive and rational, and will modify their strategies or the information structure in the task environment to maximize information returns from information-seeking activities. Two concepts in Information Foraging Theory, namely *Information Patch* and *Information Scent*, are used in the models Web User Flow by Information Scent (WUFIS; Chi, Pirolli, Chen & Pitkow, 2001), Scent-based Navigation and Information Foraging in the ACT cognitive architecture 1.0 (SNIF-ACT 1.0; Pirolli & Fu, 2003) and SNIF-ACT 2.0 (Fu & Pirolli, 2007), to explain and predict goal-directed exploration of websites. In these models of Web exploration, information patch refers to individual webpages, sub-sites of webpages, or entire websites, between which users navigate. Information

scent refers to the assessment of proximal cues, such as the text label of a link on the current webpage, on how likely the link will lead to webpages with information that satisfy the task. Information scent drives label-following behavior.

WUFIS (Chi et al., 2001) models the probability distribution of users following each link on a webpage. These probabilities are based on the information scent of each link given the goal description, and are computed using Term Frequency by Inverse Document Frequency (TF.IDF), a common technique in statistical natural language processing. WUFIS represents these webpages, links and probabilities in a matrix, represents the goal in a vector, and simulates the flow of users through the website by iterative matrix multiplication. The matrix multiplication over entire webpages of links implements a global decision that optimally flow users through all links based on their information scents. The outcome is a probability distribution of the webpages users are likely to end up in the website. However, its matrix representation and multiplication approach do not capture or consider the layout position of links on a webpage (Table 1: Layout Position), thus, may have the same problem as AutoCWW where model predictions may not match participant data at a more detailed level of analysis when layout matters. WUFIS is the underlying prediction model in the tool Bloodhound and Section 2.6 describes Bloodhound in detail.

SNIF-ACT 1.0 (Pirolli & Fu, 2003) is the first of two process models of Web navigation based on Information Foraging Theory and implemented in the ACT-R cognitive architecture (Anderson & Lebiere, 1998; Anderson et al., 2004). SNIF-ACT 1.0 models the process of a user visiting a webpage, evaluating the links on the webpage, selecting a link to go to a new webpage, or going back to a previous webpage. On visiting a webpage, the infoscent of each link is computed using Pointwise Mutual Information (PMI), as an approximation to the spreading activation in human declarative memory when assessing inter-word relatedness. Each link on the webpage is represented by a production rule that if fired will select that link. All rules then compete and the rule with the highest utility (i.e. infoscent) fires. Through this process, SNIF-ACT 1.0's assumes that all the links on a webpage get evaluated before the decision to globally select the best link (Table 1: Modeling Approach). The model does not capture the layout position of links, thus link selections are not influenced by layout position (Table 1: Layout Position). Fu and Pirolli (2007) further analyzed participant performance data and found that while the infoscent of a link predicted participants' choices better than the on-screen position of the link, participants did tend to select links located at the top of the webpage over those located at the bottom of the webpage.

SNIF-ACT 2.0 (Fu & Pirolli, 2007) removes the assumption that all links get evaluated. In its place, SNIF-ACT 2.0 serially evaluates each link on the webpage and uses a Bayesian Satisficing process that adaptively decides when to stop evaluating links on a webpage and select the best one so far. This local decision to stop or continue is not fixed, but is dependent on the infoscent of the links that have been evaluated so far (Table 1: Modeling Approach). However in SNIF-ACT 2.0, the links on the webpage are encoded left-right then top-down into a linear list which the model evaluates serially. This ordering may not be appropriate for other UI layouts (Table 1: Layout Position, note [f]). Fu and Pirolli (2007) noted that "SNIF-ACT was developed at a level of abstraction that was not sensitive to different visual layouts of the webpages" (p. 400) and that a "theory of attention

allocation as a function of different visual layouts is definitely important in predicting navigational behavior" (p. 400).

2.3.4 MESA

Method for Evaluating Site Architectures (MESA; Miller & Remington, 2004) models the interaction between website architecture (the number of links per webpage and the depth of the website), quality of links (infoscent) and human cognition limitations (serial evaluation of links due to visual attention and limited working memory). Like SNIF-ACT 2.0, MESA simulates the exploration process one step at a time. MESA is constrained to focus on and evaluate one link at a time to reflect the limitation of human visual attention. MESA uses a threshold strategy where the model immediately selects the first encountered link with an infoscent that exceeds a fixed threshold value. The threshold strategy is combined with an opportunistic strategy that will lower the threshold value if there are no links on the webpage that exceeds the original threshold value. The opportunistic strategy will then scan the webpage again and select the first encountered link that exceeds the lower threshold value, or if none exceeds the threshold value, the model will back up to the parent webpage. To reflect limited human working memory, MESA does not remember all the threshold values it used as it traverses the webpages. This means that after the threshold value is lowered, the model will henceforth select less relevant links as links get evaluated.

MESA's serial evaluation process and cognitive limitations mean that later link selections and webpage visits depend on the links and webpages already visited. The current threshold value depends on what other webpages have already been visited on the exploration path and a lowered threshold value may result in additional links being selected. Miller and Remington specifically contrasted this local decision process of MESA to the global decision process of WUFIS (Table 1: Modeling Approach, note [d]). However, MESA's focus is on site structure and not on the structure or layout of links in a webpage. MESA's representation of the UI has a fixed order in which links get evaluated for each run. Miller and Remington (2004) explained that "MESA is neutral as to the actual order in which the links are evaluated" (p. 233) and that "MESA's representation establishes a fixed order in which links are evaluated for each run. For our simulations, we can remove the effect of order by randomly ordering links for each run and then taking performance averages across many runs" (p. 234). The authors further noted that they "have not considered the effect that grouping or ordering links has on navigation times" (Miller & Remington, 2004, p. 261) (Table 1: Grouping and Layout Position). Like AutoCWW and WUFIS, MESA does not capture or consider the layout position of links on a webpage, thus, may have the same problem as AutoCWW where model predictions may not match participant data at a more detailed level of analysis when layout matters.

2.3.5 Exploratory Act and Normalization Assumption

Young (1998) and Young and Cox (2000) presented a rational analysis of exploratory choice framework that uses the concept of Exploratory Act (EA) to account for both *free exploration* to learn about the device, where the efficiency of the EA is defined to be the information gain (ΔI) per unit cost (C) of the act, and *focused exploration* to complete a particular goal, where the efficiency of the EA is defined to be the probability (P) that it will lead to the goal (G) minus the cost (C) of getting to the goal. At each step in the exploration, the efficiencies of all the EAs possible at that

moment are calculated and the EA with the highest efficiency gets selected and executed. This framework provides for both free and focused exploration to happen in an interleaved fashion. However, normalization between the two different efficiency computations was unresolved (Young & Cox; 2000) and not implemented (Cox & Young, 2004).

A key assumption in the framework is that when an EA makes an assessment of the likelihood (i.e. infoscent) that an option will lead to the goal, the assessment is dependent on all the other assessments on other options that have been made. This *normalization* assumption implies that the infoscent of on-screen options that have been evaluated will affect the decision of whether to select an on-screen option or to continue to assess another option. While the normalization assumption is different from the Bayesian Satisficing processs in SNIF-ACT 2.0, the observable outcome from these two processes is that the models adaptively decides to continue evaluate another option or stop evaluating options based on the infoscent of options that have been evaluated so far. Thus, like SNIF-ACT 2.0, the exploration process is serial and based on local decisions.

Based on this framework and its normalization assumption, Cox and Young (2004) and Brumby and Howes (2004) developed models to explain observed behavior in an experiment on menu exploration (Brumby & Howes, 2003), where participants were given an information search goal and asked to search a vertical menu of 16 items with 1 correct item and 15 distracters. Cox and Young's model is implemented as a LISP program outside the confines of a cognitive architecture. The normalization assumption is implemented by simply normalizing each infoscent estimate over the sum of all estimates after an assessment or reassessment of a menu item. Brumby and Howes' model is implemented as an ACT-R model. Its normalization assumption is implemented using ACT-R's spreading activation mechanism by (1) having a slot for each menu item in its ACT-R goal chunk, (2) updating the slot when its corresponding menu item gets evaluated and (3) utilizing ACT-R's sources of activation mechanism to "share" a fixed total amount of source activation through these slots to all declarative knowledge chunks associated with the menu items. The higher the activation of those knowledge chunks associated with a menu item, the higher the item's infoscent. Normalization takes place because the total amount of source activation is fixed.

However, in both Cox's and Brumby's models, the model is pre-configured with the number of items in the menu and that number is essential for the computations involved in the normalization. The authors did not offer a psychologically plausible explanation for how the model, and likewise the user, would know this number beforehand.

2.3.6 DOI-ACT

To investigate the interaction between infoscent and more complex on-screen layouts, Budiu and Pirolli (2007) developed DOI-ACT, an ACT-R model of navigation in degree-of-interest (DOI) trees (Figure 4). Compared to the linear lists of options and the predominant top-to-bottom order of evaluation in SNIF-ACT 2.0, Cox's and Brumby's models, the DOI tree lays out options (or nodes) in both dimensions on screen. The DOI-ACT model may attend to any group of nodes on the screen, although it still evaluates nodes within a group in a serial top-to-bottom order (Table 1: Layout Position, note [i]). DOI-ACT has two main components: (1) a visual search component that parses the screen into visual groups and selects the most salient one to attend next, and (2) a semantic

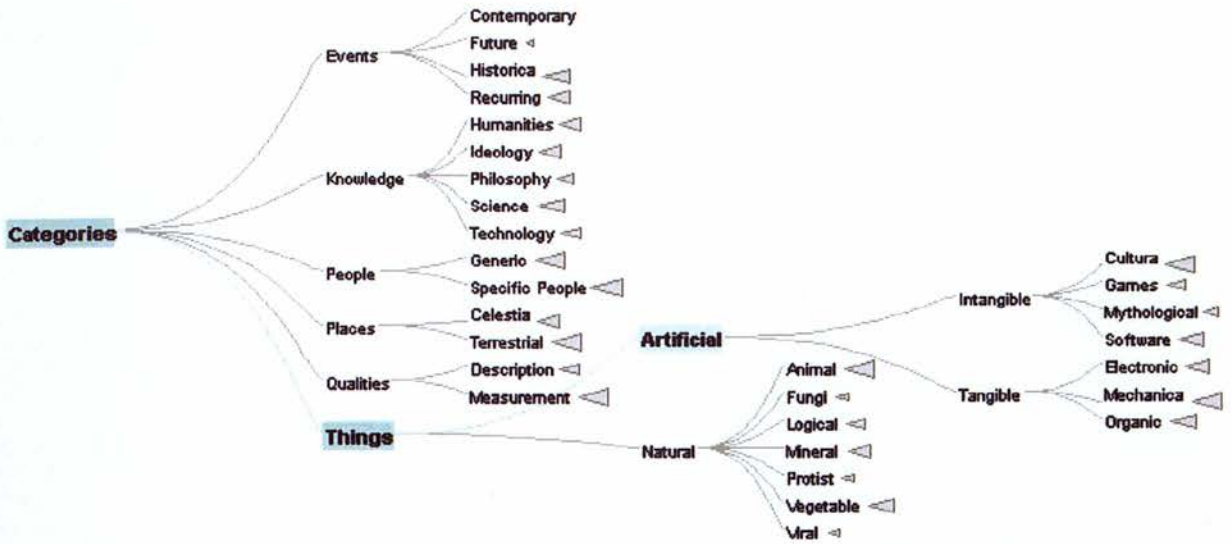


Figure 4: DOI-Tree Visualization (Budiu & Pirolli, 2007)

component that examines the nodes in the most salient visual group and decides on which one to click. To better reflect the taxonomical organization of information in the DOI tree, DOI-ACT uses two measures of infoscent: category scent and similarity scent. Category scent measures how much the search goal is a member of the class denoted by the label of a node. Budiu and Pirolli collected category scent scores from human category ratings of 1760 word pairs via a Web questionnaire. Similarity scent is computed using PMI, as is the case in the SNIF-ACT models (Table 1: Information Scent, note [h]).

Each time the model wants to attend to a new visual group, a parallel process calculates the visual salience of each visual group on-screen and the one with the highest visual salience is selected. The visual salience function is composed of the following factors:

- Horizontal distance (D) between the center of the group and the node last clicked
- Number of descendants (N) of the node last clicked that are within the group
- Category scent (S) defined as either an average of all category scents of previously visited nodes in the group, or, if no nodes were visited, the maximum category scent of all the parents for all the nodes in the group, and
- Inhibition factor (I) to reduce the salience of items that have been clicked recently so that the model has knowledge of what had been already visited and tend not to select the same groups over and over again

The horizontal distance (D) factor in the salience function means that a group that is further away to the right side (positive direction) of the screen is more attractive. Budiu and Pirolli (2007) noted that participants, "once they are on the right path, using distance as the main factor is a strategy that optimizes the time to the solution (the farthest away descendants of the current node would need to be clicked to get to the solution most quickly)" (p. 848). While this may be appropriate for

the DOI tree layout in Budiu and Pirolli (2007), it may not be appropriate for other UI layouts in general (Table 1: Visual Search, note [g]). Section 2.4.1 reviews eye-tracking experiments on other types of UI layout that found participants tended to fixate and attend on options nearest to the current point of visual attention.

2.3.7 Research Progression in Prior Theories and Models

As shown in Table 1: Scope of Exploratory Behavior, pioneering research (CE+, IDXL, LICAI+/CoLiDeS) seek to provide an account for a range of user exploratory behaviors, but may have been challenged to implement all the behaviors into a fully executable model (Table 1: Model Implementation). Evaluation was by qualitative comparison of the model to observed patterns of exploratory behavior. Subsequent research focused on label-following and visual search, and all were implemented as executable models. Evaluation of the later models was by quantitative comparisons of predictions from model runs to observed participant data.

All the models reviewed have label-following driven by infoscent, and successive research work have identified different components that make up infoscent (Table 1: Label Following). Some related work used human ratings or had the modeler assign the infoscent values in the models, which was useful and sufficed for the development and testing of theory. Other related work developed and used computational methods, such as LSA and PMI, to automatically generate an engineering approximation of infoscent, which has been shown to predict label-following behavior (Table 1: Information Scent). The use of methods like LSA and PMI remove the assignment of infoscent values as a free parameter in the model, and have the potential to make tools like Bloodhound and AutoCWW usable for practitioners (Table 1: Tool for Practitioner).

All the models reviewed use one of two modeling approaches, either a global evaluation of all available options and selection of the globally optimal choice, or a serial evaluation of available options and selection by local decision that may not be globally optimal (Table 1: Modeling Approach). The latter approach reflects the constraint of visual attention and the lower level visual search process during exploration, and is used by all the models that consider UI layout position (Table 1: Layout Position). In these models, the serial processing of options is accompanied by mechanisms to stop and make a selection (e.g. SNIF-ACT 2.0's Bayesian Satisficing, MESA's threshold value and Young's Normalization Assumption in Cox's and Brumby's models). The serial evaluation of options may not necessarily evaluate all available options before a selection is made, therefore the order in which options get evaluated directly affects the selections made during exploration.

The order of evaluation is dependent on visual search which is influenced by the layout and other visual properties of the UI, and by prior knowledge, strategies or preferences that the person may have. SNIF-ACT 2.0 and Cox's model include UI layout information but the information consist of a linear list of options in their on-screen spatial ordering (i.e. before-after) instead of their actual on-screen positions (Table 1: Layout Position). Brumby's model includes on-screen positions but the UI layout was a relatively simple one-dimensional vertical list of menu items. DOI-ACT has an accurate representation of the two-dimensional DOI tree UI but its visual search process assumes knowledge specific to the DOI tree UI layout and may not transfer to other UI layouts (Table 1:

Visual Search, note [g]). The next two sections will review relevant work in visual search and UI device models.

2.4 Visual Search

Psychological research on visual search (see Wolfe, 1998 for a review) has identified numerous basic features (e.g. color, orientation, size and spatial frequency) that are available pre-attentively. While simple search targets could be identified pre-attentively (referred to as *parallel search* or *efficient search*), more complex targets with a conjunction of features have to be searched for and examined under attention (referred to as *serial search* or *inefficient search*). Pre-attentive bottom-up processing (stimulus-driven local differences in features that "pop-out" a visual object, and local similarities in features that group visual objects) and top-down processing (user-driven control to limit features and values, such as a certain color or orientation) guide the subsequent deployment of attention in visual search.

2.4.1 Eccentricity and Inhibition of Return

A factor that influences the deployment of visual attention is eccentricity. Targets take longer to locate as their distance from the current point of visual attention increases, which may be due to the decline of visual acuity in the periphery, and that it is simply a longer distance for the eye to travel. Assuming there is no privileged knowledge favoring certain on-screen objects (as there was in the case of DOI trees discussed in Section 2.3.6), the most efficient strategy will be to make the shortest possible shift. Rational human behavior to optimize search efficiency will tend to next look at nearby objects rather than objects further afield.

Halverson and Hornof (2006; 2007) modeled the eye-tracking data of participants searching text labels in a two-dimensional layout on screen. Instead of following a prescribed visual search path as was in their earlier model (Hornof & Halverson, 2003; Hornof, 2004), the new model attends to the next label with the least eccentricity from its current point of visual attention. To account for human variability, a fluctuation factor (i.e. noise) is applied when the eccentricity of each on-screen object gets updated for a new eye position. Fleetwood and Byrne (2006) modeled the eye-tracking data of participants searching graphical icons in a two-dimensional layout on screen similar to those found in graphical UIs. Their visual search model shifts attention to the next candidate icon nearest to the icon that is the current focus of visual attention, as was observed in their participants' eye-tracking data.

Another factor that influences the deployment of attention is called "inhibition of return" in the visual search literature. Although empirical evidence for inhibition of return has been mixed and is not well understood, Wolfe (1998) argued that "visual search models with a serial search component need to ask how attention 'knows' where it has been" (p. 55) and "there must be some way to keep track of the loci and/or objects that have been examined and rejected in the course of a search" (p. 55). Analysis of eye-tracking data of participants searching text labels (Halverson & Hornof, 2006; 2007) and graphical icons (Fleetwood & Byrne, 2006) showed that participants rarely fixate on an object more than once. Their visual search models to explain participant data implement inhibition of return: the visual search proceeds without replacement, i.e after an on-screen object has been visually attended and fully identified it will not be attended again.

2.4.2 GROUPING

The pre-attentive, bottom-up stimulus-driven processing of similarities in local features group visual objects. Wolfe (1998) noted that "several theories of search rely on grouping mechanisms to make conjunction search more efficient" (p. 51) and "most grouping accounts suggest that search can be speeded by processing and rejecting distracters in groups rather than one at a time" (p. 51). Wolfe (1998) concluded that "probably a truly satisfactory model of search will need low level grouping in addition to top-down and bottom-up selection processes" (p. 52).

From analyzing eye-tracking data of participants searching graphical icons, Fleetwood and Byrne (2006) suggested that pre-attentive visual features like icon color and shape made up groups of icons, and participants constrained their search to within a group of icons that shared common features with the target icon, before moving to a further group that also shared common features with the target icon. Their ACT-R model of icon search uses the ACT-R vision module to pre-attentively identify candidate icons that share features with the target icon. The group-based search behavior emerges from the model shifting attention to the next candidate icon nearest to the icon that is the current focus of visual attention, i.e. eccentricity coupled with common visual features led to group-based visual search.

Hornof and Halverson (2003) had participants search for text labels in both labeled and unlabeled groups. Groups were spatially demarcated on screen and labeled groups had an additional on-screen label next to the group. All labels were either three-letter words (e.g. BEG, MAX, RED) or three-letter pseudo-words (e.g. VIN, KEZ, ZIL) and randomly assigned to groups. In both conditions, participants were given the exact target label to search for, and in the labeled group condition, participants were also given the exact label of the group where the target label resided. Participants completed the search tasks in the labeled group condition faster than in the unlabeled group condition. The authors suggested that participants knew or learned to use the strategy of first searching for the given group label and then the target text label. Hornof's (2004) model of these visual search tasks employs such a strategy.

Hornof (2004) explained that semantics was removed from the above text label search tasks by using those three-letter words and pseudo-words, and by randomly assigning labels on screen and into groups. Halverson and Hornof (2008) began to investigate the effect of semantic grouping on visual search by using meaningful group and text labels (e.g. "jewelry", "anklet", "bracelet", "cufflink"), varying the semantic cohesiveness of text labels within a group (cohesive versus non-cohesive) and varying between using labeled versus unlabeled groups (see Figure 5). Participants were given the exact target label to search for but were not given the label of the group where the target label resided. The main result was that participants completed the search tasks faster when groups were cohesive. Halverson and Hornof suggested that participants may judge the semantic relevance of a cohesive group after evaluating one or a few labels in the group, and that enabled participants to discount and skip a group if the group was not semantically likely to contain the target label. In the non-cohesive group condition, it was not possible to discount a group in that way and participants had to evaluate the labels more exhaustively.

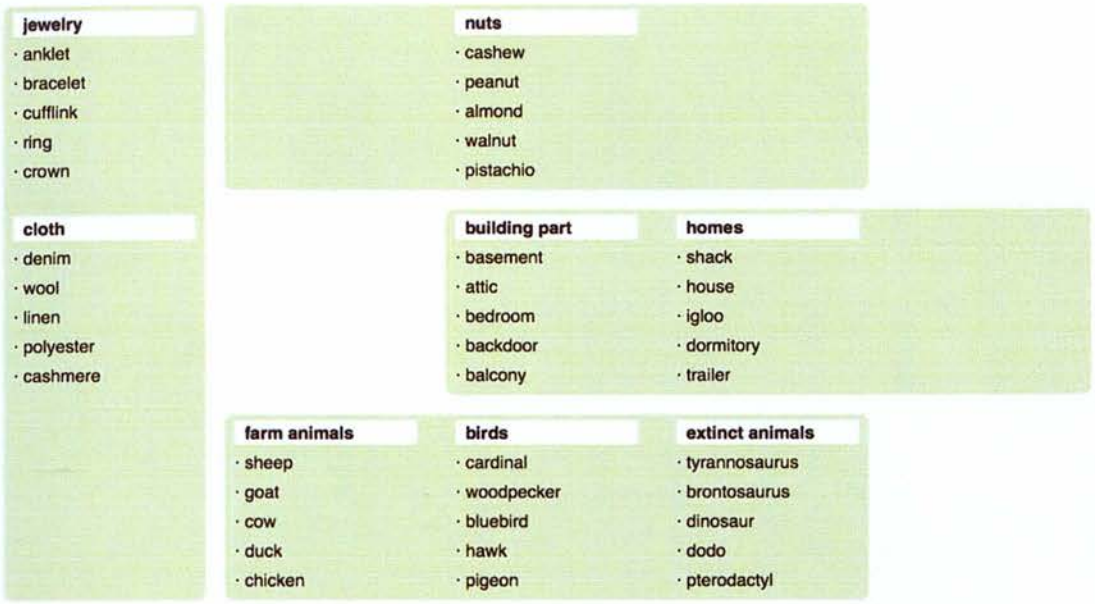


Figure 5: An example layout with semantically cohesive groups, group labels and background color in Halverson and Hornof (2008)

2.4.3 Visual Search in User Exploration

Prior research has provided insight into the visual features and processes that guide visual search. Eccentricity, inhibition-of-return and grouping have been implemented in models and model predictions had good fits to eye-tracking data. However, most prior research had participants searching for exact known targets, which is less likely to be the case if the user is unfamiliar with the UI and exploring. Experimental results from Blackmon et al. (2005) and Halverson and Hornof (2008) showed that semantic grouping affects exploration task performance. More research is needed, but it is evident that both the positions and semantics of individual options, and the positions and semantics arising from organizing structures on screen, have an effect on exploration.

For the visual search process to correctly consider layout position and grouping, the visual search must take place on an accurate representation of the UI being explored. The next section reviews research on UI device models.

2.5 Device Models

There are two approaches in some prior research work: the human modeler (1) writes a software program that replicates the behavior of the UI being explored, or (2) fills in software data structures with information about the UI being explored. An example of the software program approach is IDXL, which implemented "a minimal simulation of the Cricket Graph interface with which the model interacts" (Rieman et al., 1996, p. 756). The simulation reports to the model the menu item that is at a particular location in the menu system whenever the model directs its attention to that location. The simulation also responds to actions by the model, such as the selection of a menu item, and changes the simulation state to reflect that a sub-menu has appeared. An example of the data structure approach is LICAI+, where the modeler writes propositions for each on-screen object in the Cricket Graph interface to capture information like text label, action afforded, etc. These propositions are then included in the integration process of LICAI+. These two approaches in IDXL and LICAI+ provide the modeler with complete control over the device model. The modeler can directly change the device model by rewriting software code or changing data values, which is convenient for modeling research. However, the software program approach requires sufficient knowledge of an additional programming or modeling language to create the simulation, and both approaches, especially the data structure approach, involve much manual effort on the part of the human modeler; both approaches are not practical for practitioner use.

A third approach is to provide the process model "direct" access to the actual UI, by automatically translating the output from the actual UI into a representation accessible by the model, and translating the actions of the model into input on the actual UI. Simulated Hands and Eyes (SHE; Misker, Taatgen & Aasman, 2001) and Segmentation/Manipulation (SegMan; Amant, Riedl, Ritter & Reifers, 2005) are two examples of this approach. The long-term goal is for the model to interact with the same actual UI that people interact with. Practitioners can then bring an actual software or device they want to test to the model and use the model like a "black-box". However, SHE depends on software hooks into the actual UI and only works for software written in certain programming languages. SegMan uses image-processing techniques on screen-captures of the actual UI but sometimes failed to translate parts of the UI.

A fourth approach strikes a middle-ground between the first two approaches and the third approach. The process model interacts with a simulated UI or access data records that represent the UI, but the simulated UI or data records are constructed automatically or semi-automatically prior to the running of the model. Bloodhound, the practitioner tool implementation of WUFIS (discussed in Section 2.6), automatically fills up its data matrix by crawling and parsing webpages from a given website. In CogTool, the practitioner can mock up the UI design by dragging and dropping standard UI widgets, such as buttons, menus and links, from a palette of widgets onto frames (Figure 6). A frame represents a display state of the UI, and interface actions such as a mouse button press or click on a widget can be specified by drawing transitions from that widget in its frame to another frame. CogTool automatically creates an ACT-R device model from a mockup of a UI design. The ACT-R device model is a program that simulates the UI that an ACT-R model can look at and interact with.

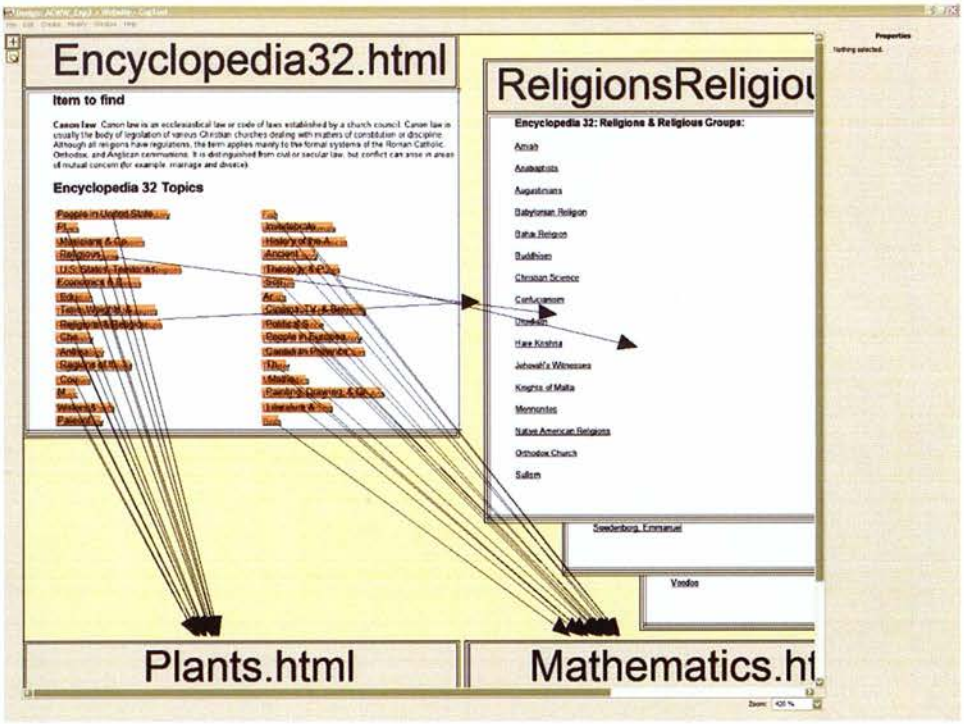


Figure 6: Mockup of a website UI design in CogTool. CogTool can convert the mockup into an ACT-R device model that an ACT-R model can interact with.

The first two approaches may be more efficient and work well in modeling research for theory development like in IDXL and LICAI+, but are too difficult and time-consuming for practitioners. The third approach is an attractive goal in the long term, but the automated translation currently either only works for some UIs or may mistranslate parts of the UI, thus, more research is needed. The third approach also makes testing UI redesigns potentially difficult as it entails modifications to the UI of the actual software or device. The fourth approach requires less specialized technical knowledge and effort compared to the first two approaches, and provides the modeler or practitioner the option to inspect and check the device model for correctness prior to running the model. In the case of CogTool, this approach makes it much easier to create and test UI redesigns by modifying the UI mockup instead of the actual software or device.

2.6 Modeling Tools

Automated Cognitive Walkthrough for the Web (AutoCWW; Blackmon et al., 2005) is a publicly available tool accessible on the Web[3] to help practitioners analyze one or more webpages for difficulties users may face during exploration of a website. AutoCWW draws its theoretical basis from CoLiDeS and uses LSA to compute engineering approximations of semantic similarity and familiarity (Table 1: Modeling Approach, note [e], and Information Scent). To analyze a webpage, the practitioner manually enters the text description of likely exploration goals, the heading labels of all sub-regions on the webpage and the text labels of all links on the webpage. The analyst then flags the correct link or links for each goal, sets up one or multiple analyses with different sets of parameters (e.g. different LSA semantic spaces for users of different reading levels, different text label elaborations, etc) and submits the analysis request to AutoCWW. AutoCWW analyzes the submitted goals, headings and links for potential problems by computing the LSA cosine, its measure of semantic similarity, between the text of the goal description and the text of each heading, and between the text of the goal description and the text of each link. AutoCWW also computes the LSA term vector length and frequency counts of the heading texts and link texts, to evaluate if the user, as represented by the selected LSA semantic space, has an adequate level of relevant background knowledge to be familiar with the headings and links (see Section 2.3.2 on CoLiDeS' five factors that combine to measure the attractiveness of a label).

AutoCWW reports if the correct link or links on the webpage might be unfamiliar to the user (Unfamiliar Correct Link) or might be too semantically different from the goal to be chosen (Weak-Scent Correct Link). AutoCWW reports if there are incorrect links that are semantically related to the goal and thus compete with the correct link. By analyzing the headings of sub-regions, these competing links may be nested under a correct or incorrect heading that is semantically related to the goal (Competing Link under Correct Heading *or* Competing Link under Competing Heading). AutoCWW regards a search goal that results in any link being reported with one or more of these problems as a goal With Problems, where users are more likely to have difficulties during exploration. From the number and types of problematic links identified on a webpage for a particular search goal, AutoCWW predicts the mean number of link clicks users will make to select a correct link on that webpage for that search goal. AutoCWW assembles the analysis results in Microsoft Excel spreadsheets and emails the spreadsheets to the practitioner.

Bloodhound (Chi et al., 2003) is a Web-based tool to predict the percentage of users that will reach the target webpage and be successful in exploration, and the ranking of webpages that users are most likely to visit when exploring a website with a search goal. Unlike AutoCWW, Bloodhound[4] is not publicly available but is available for licensing. Bloodhound provides a Web-based UI for practitioners to run WUFIS, which models the probability distribution of an arbitrary number of users following each link on a webpage (Section 2.3.3 describes how WUFIS works). To use Bloodhound, the analyst submits the Web address of the webpage where exploration starts

[3] Accessed on January 17, 2011 at http://autocww.colorado.edu/~brownr/ACWW.php

[4] Retrieved on January 17, 2011 from
http://www2.parc.com/istl/groups/uir/projects/bloodhound/bloodhound.htm

(exploration is limited to the webpages under the same domain as the starting Web address), the keywords of one or multiple search tasks and the Web address of the target webpage for each of the tasks. Bloodhound uses this information to setup the data matrix in WUFIS and runs the model. Bloodhound then displays a webpage that reports the average success rate (percentage of users predicted to find the target webpages) over all the search tasks, the success rate for each task, and the webpages that were most often visited over all the search tasks.

Both AutoCWW and Bloodhound provide a UI for practitioners to enter the necessary information to setup an analysis or model run, and generate predictions of exploration task performance. Both return their predictions in a format that is accessible to the practitioner: spreadsheets from AutoCWW and a webpage of results from Bloodhound. Besides AutoCWW and Bloodhound, the rest of the models reviewed were not further developed into tools for practitioner use (Table 1: Tool for Practitioner). Setting up, running and extracting prediction results from those models typically involve software programming, manipulating data formats and managing multiple files, activities that are tedious but acceptable for a modeling researcher, but not practical for a HCI practitioner.

3 RESEARCH GAPS

Table 1 summarizes the review of prior related work presented in Section 2. Inspecting the table reveals seven research gaps as indicated by the seven columns with a majority of white cells. There is a lack of further research on the Goal Formation, Learning and Interface Action aspects of exploratory behavior (see these columns in Table 1) after LICAI+/CoLiDeS. This dissertation does not seek to address these three research gaps, which are orthogonal to the research focus of this dissertation and can be topics for future work. The next research gap in Visual Search ties in with and is "manifested" through the two research gaps in Layout Position and Grouping, in terms of how visual search over layout position and grouping on screen affect user exploration. Finally, there is the research gap from the scarcity of work in developing Tools for Practitioners.

Table 1 also reveals that automated methods to compute the infoscent scores (present in 7 out of 13, Table 1: Information Scent) that drive label-following (present in 13 of 13, Table 1: Label-Following) have been progressively developed and successfully used in a number of prior models and tools. This dissertation does not seek to advance the state-of-the-art in algorithms to compute the infoscent scores that drive label-following. Instead, this dissertation will utilize a proven method like LSA to control for the semantic component of the model while developing the visual search, layout position and grouping components of the model.

This dissertation focuses on the research gaps in Visual Search, Layout Position, Grouping and Tool for Practitioner. Sections 3.1 and 3.2 will address Layout Position and Grouping in conjunction with Visual Search. Section 3.3 will address Tool for Practitioner.

3.1 CONSIDERATION OF LAYOUT POSITION

Only 4 out of the 13 prior related works consider the effect of layout position in conjunction with visual search on user exploration (Table 1: Visual Search and Layout Position). Visual search over layout position in IDXL and SNIF-ACT 2.0 serially evaluates on-screen options based on their spatial ordering (i.e. before-after) instead of their actual on-screen positions. Brumby's model uses on-screen positions but the UI layout was a relatively simple one-dimensional vertical list of menu items. DOI-ACT includes a more complex two-dimensional UI layout but its visual search assumes knowledge specific to the DOI tree UI layout and prefers to shift visual attention further afield to the right, which is different from the eccentricity heuristic favored by other models of visual search (Fleetwood & Byrne, 2006; Halverson & Hornof, 2006; 2007; Halverson, 2008) and may not be appropriate for other UI layouts in general. DOI-ACT further assumes top-down visual search over the nodes within a group. In modeling tools, both Bloodhound and AutoCWW do not capture the layout of the UI. Their global evaluation processes, inherited from WUFIS and CoLiDeS respectively, are neutral to the order in which a person would evaluate the options on the UI.

However, the layout position of options in a UI may affect the exploration choices actually made, because a user is not likely to choose an option that he or she did not look at and evaluate, and competing options seen before the correct option might be chosen instead. Figure 7 illustrates this with an actual webpage from AutoCWW Experiment 2 (Blackmon et al., 2002), and a modified version of the same webpage. Assuming a predominant left-to-right visual scan pattern, the expectation is that participants would be more likely to click on the correct link if it appeared in the

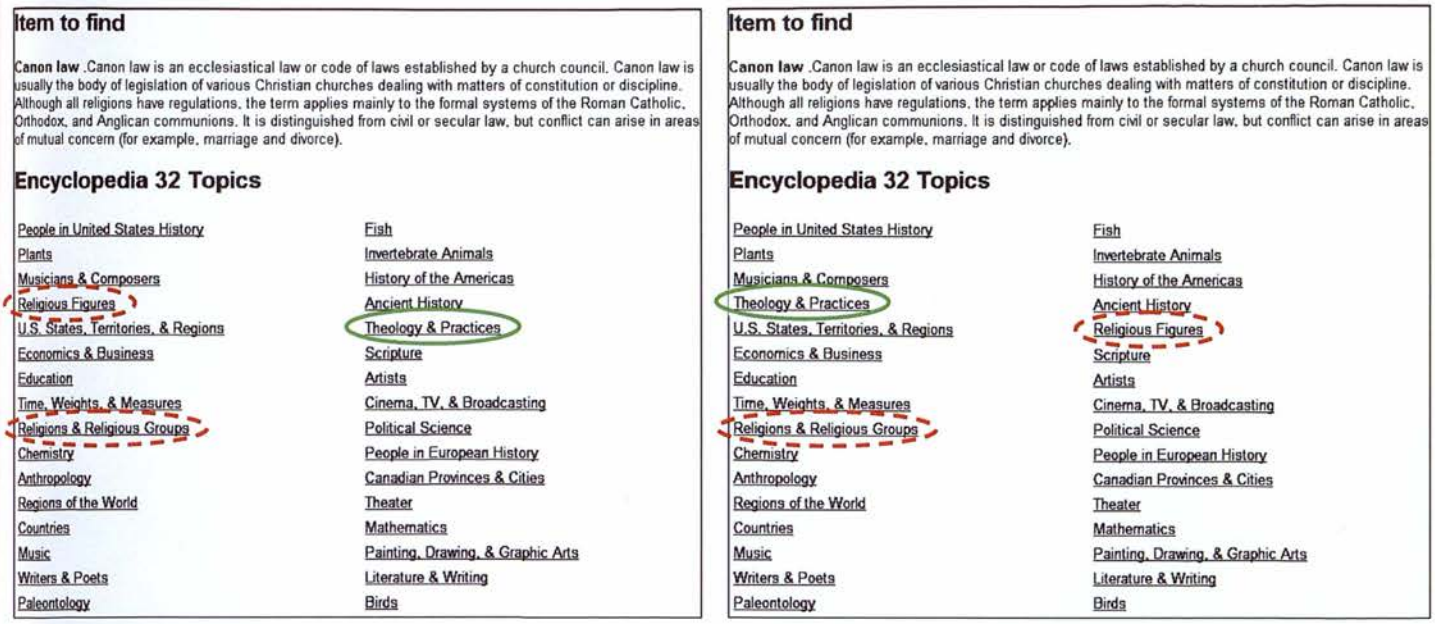


Figure 7: A search goal and webpage used in AutoCWW Experiment 2 (Blackmon et al., 2002). Each search goal had one correct link on the webpage. The correct link for the goal "Canon law" is "Theology & Practices". AutoCWW identified links "Religious Figures" and "Religions & Religious Groups" as competing links that are also semantically related to the goal. The left picture shows the original layout. The right picture shows a modified layout with links "Theology & Practices" and "Religious Figures" swapped. The hypothesis is that users would be more likely to click on the correct link in the modified layout than in the original layout.

left column than if it appeared in the right column. If true, this is significant because a user clicking on an incorrect link can increase the number of interaction steps and time spent exploring the wrong branches in a large and complex website.

To investigate the effect of layout position, a further analysis of AutoCWW Experiment 2 was done as part of this dissertation (Teo and John, 2008, reproduced in detail here). The experiment webpages are publicly accessible on the Web and Dr. Marilyn Blackmon generously shared the participant log files with us. The experiment had 64 search tasks, 32 of which were attempted on a webpage with 32 links in two 16-link columns (Figure 7). Each task was successfully performed by 22 or 23 participants. We analyzed only the 22 tasks for which AutoCWW judged the correct link to be semantically related to the goal, reasoning that if the user could not recognize that the correct link as related to the goal, then its position on the webpage would not matter.

Figure 8 shows the analysis by column position of the correct link on the webpage for the two performance measures reported in the AutoCWW experiments: the number of clicks performed on the webpage, where the last click was on the correct link (mean clicks on webpage), and the percentage of trials where the first click on the webpage was on the correct link (percent first click success). Although these measures are highly (negatively) correlated, they represent two subtly different usability concerns. Mean clicks is important if a design team has data suggesting that users will be willing to explore a bit, but will leave the website if it takes too long to find what they want. Percent first click success is important if a usability requirement is expressed as a certain

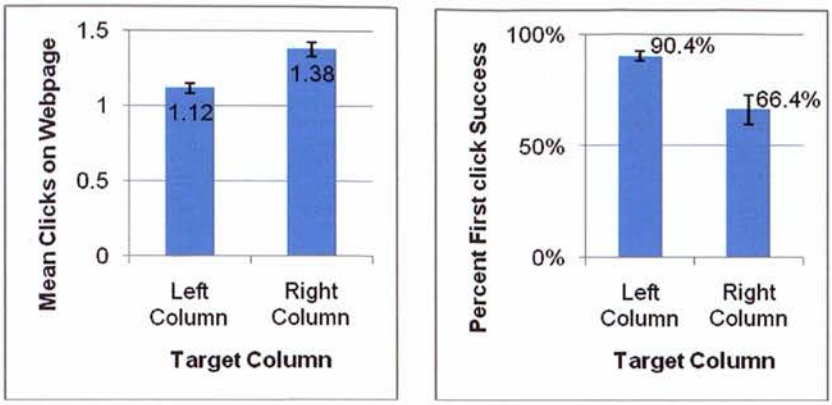


Figure 8: Participants' mean clicks on webpage and percent first click success by target column (Standard error shown)

percentage of users achieving their goals without error. Since these measures are different, we carry them through our analysis.

Participants indeed made significantly fewer clicks when the correct link was in the left column (M = 1.12 clicks, SD = 0.13) than in the right column (M = 1.38 clicks, SD = 0.15) (F (1, 20) = 19.0, p < 0.01). They also had a significantly higher percent first click success when the correct link was in the left column (M = 90.4, SD = 7.9) than in the right column (M = 66.4, SD = 19.6) (F (1, 20) = 16.0, p < 0.01). These results support our hypothesis and suggest a predominant left-to-right visual scan pattern as was also found in eye-tracking studies of visual search in similar text layouts with participants from a similar culture (Halverson & Hornof, 2006; 2007).

We wanted to find out if existing models would be able to predict this effect of layout position on participant task performance. The obvious choices were the two modeling tools, AutoCWW and Bloodhound, as that meant we could readily generate predictions from these models as intended by their authors. Unfortunately, Bloodhound[5] is not publicly available. There are 5 other prior related works that consider layout (Table 1: Layout Position), however, IDXL is not a fully executable model; both Cox's and Brumby's models performed exploration in a one-dimensional list of menu items but did not describe how their models should explore a two-dimensional layout; the details of the algorithm that determines and pre-encodes the sequence of link evaluations in SNIF-ACT 2.0 were not available; and DOI-ACT assumes knowledge specific to the DOI tree UI layout and uses human ratings for category scent scores. Thus, we could not readily create and run these models to generate predictions. Although AutoCWW does not consider layout position, we went ahead and submitted these tasks to AutoCWW and compared its predicted mean clicks on webpage by column position of the correct link.

We entered the paragraph of text under the line "Item to find" as the goal statement (see top of Figure 7), and entered the 2-column webpage of 32 links as 16 links under 2 sub-regions. AutoCWW is designed to work with regions that have heading text, but the columns did not have heading text. Therefore, we entered the goal text as the heading text for both columns, thus, both

[5] Retrieved on January 17, 2011 from
http://www2.parc.com/istl/groups/uir/projects/bloodhound/bloodhound.htm

columns were judged by AutoCWW as being related to the goal and all links were eligible to compete. We set AutoCWW to use the "General_Reading_up_to_1st_year_college (300 factors)" semantic space, and then set AutoCWW to do the default full elaboration on the link texts because the original link texts are short, but to do no elaboration on the heading texts because the columns are not semantic or categorical groupings of the links. For each task, AutoCWW predicted the task's mean clicks on webpage.

As expected, AutoCWW did not predict any significant difference between search tasks with the correct link in the left column compared to in the right column ($F(1, 60) = 0.16$, $p > 0.05$). There are two reasons why AutoCWW did not predict the difference in participants' performance. The first is that the information requested by AutoCWW about sub-regions and links within sub-regions does not contain any layout position, so essential information is lacking (Table 1: Layout Position). The second reason is that the AutoCWW analysis globally evaluates all sub-regions and links base on information scent alone (Table 1: Modeling Approach, note [e]). The analysis does not consider layout position information, which is not available in the first place.

We did not submit these tasks to Bloodhound because it is not readily available. Like AutoCWW, the webpage and link information that Bloodhound captures does not contain layout position information. Bloodhound also globally evaluates all links on a webpage and flow the probability distribution of users optimally down each link based on the link's infoscent. Therefore, we also expect Bloodhound would not predict this effect.

This further analysis of AutoCWW Experiment 2 showed that layout position matters in goal-directed user exploration but is not predicted if existing models and tools lack a device model with layout position information and an evaluation process that uses layout information. To address this research gap:

> *This dissertation integrates information scent with a visual search process and an accurate representation of the UI layout, in a model of goal-directed user exploration.*

3.2 Consideration of Grouping

Groups in the UI can arise from proximity, color, density, etc. For example, experiments by Blackmon et al. (2002, 2003 and 2005) and Halverson and Hornof (2008) had groups demarcated by proximity and color (Figures 2 and 5). Analysis of experimental results by Blackmon at el. (2005), Kitajima at el. (2007) and Halverson and Hornof (2008) suggested that the infoscent and the grouping of on-screen options affect exploration task performance.

Only 3 out of the 13 prior related works consider the effect of grouping on user exploration (Table 1: Grouping). CoLiDeS and AutoCWW do consider grouping, but the groups lacked layout position information and are globally evaluated and selected based solely on highest infoscent in the UI (Table 1: Grouping and Layout Position). DOI-ACT defines groups based on the vertical proximity between nodes, considers the groups' layout positions, and uses a serial evaluation and local decision process. However as mentioned earlier, its visual search assumes knowledge specific to the DOI tree UI layout and prefers to shift visual attention further afield to the right, which is

different from the eccentricity heuristic favored by other models of visual search and may not be appropriate for other UI layouts in general.

Grouping affects user exploration and more research is needed to include groups as a factor in modeling goal-directed user exploration. The approach in this dissertation is for the human modeler to define the groups and provide the group information to the exploration model, which is the same approach taken by Blackmon et al. (2002, 2003, 2005), Halverson and Hornof (2006, 2007, 2008), and in DOI-ACT. This allows research on how groups affect user exploration to progress in parallel with other psychological research on how groups are formed and recognized. To address this research gap:

> *This dissertation integrates information scent with a hierarchical visual search process and an accurate representation of the UI layout that includes grouping, in a model of goal-directed user exploration.*

3.3 Implementation as a Tool

Only 2 out of the 13 prior related works implement the research models into tools for practitioner use (Table 1: Tool for Practitioner). One approach to speed up this process is to integrate the model into an existing modeling tool that has demonstrated success in reducing the effort required of a practitioner. One such tool is CogTool (John at el., 2004). CogTool provides facilities for a practitioner to create, save and modify a mockup of a UI design, features which are absent from AutoCWW and Bloodhound. CogTool includes a model of skilled task performance time, based on the Keystroke Level Model (KLM; Card, Moran & Newell, 1980; 1983), and implemented in the ACT-R cognitive architecture. CogTool automatically creates a KLM when the practitioner demonstrates a sequence of interaction steps on the UI mockup, and automatically inserts Mental Operators into the KLM, which has been an error-prone step for practitioners. CogTool then executes the KLM ACT-R model and presents the predicted task performance times to the practitioner in various visualizations, for the practitioner to compare between UI designs and improve the designs for better task performance.

To efficiently make the model of goal-directed user exploration available to the rest of the HCI community:

> *This dissertation implements the model of goal-directed user exploration as part of CogTool.*

The advantage of this approach compared to building a separate new tool is that the existing facilities in CogTool that are common to both the KLM and the goal-directed user exploration model, such as the ability to mockup a UI design with layout position information, do not have to be re-implemented. Tool development effort can focus on adding to CogTool the new facilities that are required by the user exploration model, such as the generation of infoscent scores and the specification of groups in the UI design. Another advantage is that the new model of goal-directed user exploration can reach out quickly to the existing group of CogTool users, and these users can benefit from the increased capabilities of CogTool to make predictions of both skilled and exploratory behavior on the same UI mockup.

For ease of reference, the new model of goal-directed user exploration developed in this dissertation and the additions made to CogTool to integrate and support the new model shall be referred to as *CogTool-Explorer*.

4 CogTool-Explorer

Section 3 presented the three research gaps in modeling goal-directed user exploration that are the focus of this dissertation: (1) consideration of layout position, (2) consideration of grouping and (3) implementation as a tool. Section 4 presents CogTool-Explorer, a solution researched and developed in this dissertation to address these gaps. In particular, Section 4.1 presents modeling work done in CogTool-Explorer 1.0 to address the first research gap in consideration of layout position. Section 4.2 presents further modeling work done in CogTool-Explorer 1.1 and CogTool-Explorer 1.2 to address the second research gap in consideration of grouping. Section 4.3 presents design and implementation work done to integrate CogTool-Explorer 1.2 into CogTool, to address the third research gap in implementation as a tool. Overall, the modeling, analysis and development work leading up to CogTool-Explorer 1.2 provide support for the thesis of this dissertation presented in Section 1.1.

Figure 9 presents an overview of CogTool-Explorer. Items in white background indicate components and external resources that existed prior to this dissertation, and are used by CogTool-

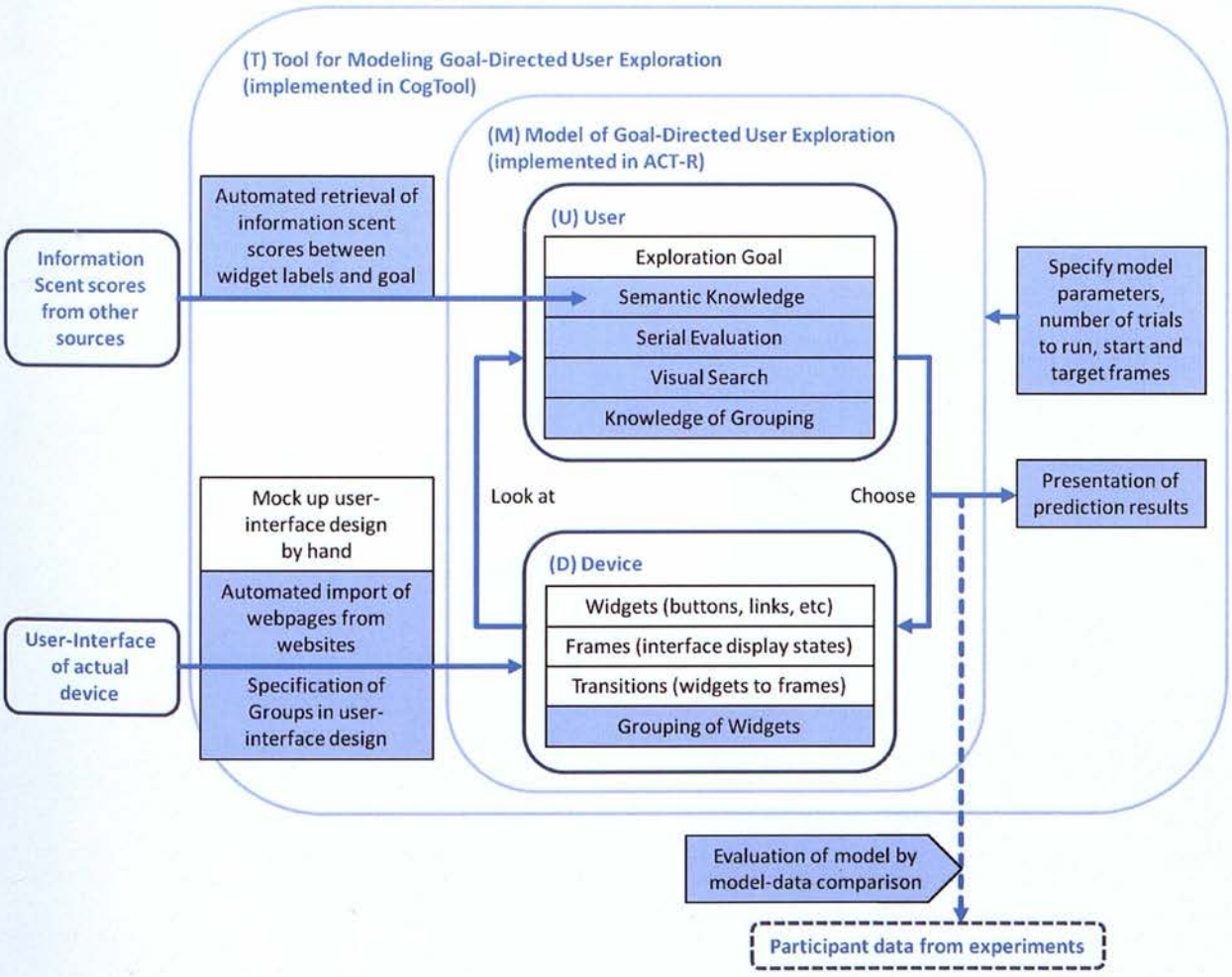


Figure 9: Overview of CogTool-Explorer. White background indicates pre-existing components and external resources. Light blue indicates new or refined components contributed by this dissertation.

Explorer. Items in blue indicate the new work that was done in this dissertation.

CogTool-Explorer consists of a model of goal-directed user exploration implemented in the ACT-R cognitive architecture (region M). The model simulates a user (region U) with an exploration goal and semantic knowledge exploring the UI of a device (region D). The model serially evaluates the infoscent of on-screen widgets in the device (in region D) guided by its visual search process and knowledge about grouping (in region U). When the model chooses a widget in the UI, the device model will update the UI with the next frame and its widgets specified by the transition in response to the interface action on the widget. This cycle continues until the exploration goal is met or time allowed for the model to run is up. This exploration takes place on a device model that accurately represents the UI of the actual device.

The CogTool-Explorer model is implemented as part of CogTool for use by practitioners (region T). In CogTool-Explorer, a practitioner can automatically or manually create the device model that represents the UI of the actual device (lower left of region T), automatically extract the text labels of widgets from the device model and retrieve infoscent scores based on the widget labels and the goal description from an external database (upper left of region T), setup the CogTool-Explorer model and specify model parameters (upper right of region T), and run the model to get predictions of likely exploration paths (lower right of region T). In the course of developing CogTool-Explorer, the model will be evaluated by comparing its predictions to data collected from participants performing the same exploration tasks.

4.1 Consideration of Layout Position

Following the analysis of participant data from the two-column layout described in Section 3.1, we developed the first CogTool-Explorer model (renamed CogTool-Explorer 1.0 in this dissertation) to consider both infoscent and layout position to make more accurate predictions of goal-directed user exploration (Teo & John, 2008, reproduced in detail below). CogTool-Explorer 1.0 integrates a serial evaluation model, with a visual search process and a UI device model that preserves layout positions. These are the three necessary components to consider layout position and CogTool-Explorer 1.0 uses them all successfully to make more accurate predictions.

4.1.1 Add Perceptual-Motor Behavior

CogTool-Explorer 1.0 uses the SNIF-ACT 2.0 model to serially evaluate links on the page one at a time. The model evaluates the link's infoscent with respect to the goal, remembers the link as the best link if it has the highest infoscent so far in the page, and then decides to either choose the best link seen so far in the page, or continues to look at and read another link. Each action is associated with an ACT-R production and the production with the higher utility is chosen. CogTool-Explorer 1.0 uses the same utility update equations as SNIF-ACT 2.0 (see 8: Utility equations in Fu & Pirolli, 2007) to update the utilities associated with these two productions every time after a link is evaluated and then decides which action to take:

ReadAnother: $$U(n+1) = \frac{U(n) + IS(link)}{1 + N(n)}$$ [Eq. 1]

ChooseBest: $$U(n+1) = \frac{U(n) + IS(Best\ Link)}{1 + k + N(n)}$$ [Eq. 2]

Fu and Pirolli (2007) explained:

> $U(n)$ represents the utility of the production at cycle n, and $U(n+1)$ represents the updated utility of the production at cycle $n+1$, $IS(link)$ represents the information scent of the current attended link, $N(n)$ represents the number of links attended on the Web page at cycle n, $IS(Best\ Link)$ is the highest information scent of the links attended on the Web page, k is a scaling parameter. (p. 380)

Since the model may not evaluate all links on a webpage before making a selection, the order in which links are evaluated has a direct effect on its predicted exploration choices. However, the original SNIF-ACT 2.0 model did not move a simulated eye and evaluate links in an order that reflected how links may be looked at in a webpage, instead it used webpage and link information collected by Bloodhound, encoded the links directly into ACT-R declarative memory chunks and "looked at" links by retrieving them from declarative memory (Fu, W.-T., personal communication, September 18, 2006). In CogTool-Explorer 1.0, we modified the SNIF-ACT 2.0 model with new model code to implement the perceptual and motor actions of looking at links on the webpage and clicking on the selected link during a model run, and modified the support code to use alternative computations of infoscent besides the original PMI function. The selection of an infoscent function, and values for several model parameters, will be discussed in Section 4.1.5 about our test of CogTool-Explorer 1.0.

4.1.2 Add a Visual Search Strategy

To guide the newly added perceptual-motor actions in CogTool-Explorer 1.0, we further modified the SNIF-ACT 2.0 model to add a visual search strategy based on the Minimal Model of Visual Search (Halverson & Hornof, 2007). The visual search strategy is implemented in the ACT-R vision module augmented with the EMMA model of visual preparation, execution and encoding (Salvucci, 2001). This strategy starts in the upper-left corner of an accurate representation of the webpage (described in Section 4.1.3) and proceeds to look at the link nearest to the model's current point of visual attention (*xy*-coordinates), where "nearest" is subjected to a fluctuation factor. CogTool-Explorer 1.0 uses the same fluctuation factor as Halverson and Hornof (2007), which is the normal distribution with a mean of 1 and a standard deviation of 0.3. CogTool-Explorer 1.0 maintains its point of visual attention when the page changes, and exhibits inhibition of return by performing the visual search without replacement, that is, on visiting a page, each link may be looked at and evaluated by the model at most once, However, in a later visit to the same page, those links may be looked at and evaluated again. Eye-tracking studies and modeling by Halverson and Hornof (2006, 2007) found that such a strategy explained for 59% of all systematic eye-scan patterns in their visual search experiments of similar text layouts.

4.1.3 Preserve Layout Position

For CogTool-Explorer 1.0 to correctly consider the order of serial evaluation, the model must interact with an accurate device model of the webpage. CogTool-Explorer 1.0 leverages the ability of CogTool to accurately represent a UI design, in particular the on-screen position, dimension and text label of every link on the webpage (Figure 6). Earlier versions of CogTool-Explorer required webpages to be mocked up by hand. To automate this process, we implemented in CogTool-Explorer 1.0 the ability to crawl and submit a list of URLs to WebRender (Reeder, Pirolli & Card, 2001), a webpage rendering tool, to render and extract the position, dimension, text and target URL of each link from the actual webpage (later versions of CogTool-Explorer, from CogTool-Explorer 1.0a onwards in Section 4.2.1.3, use the open source XULRunner instead of the proprietary WebRender, and further streamlined and automated the process; Section 4.3.1 describes this in more detail). CogTool-Explorer 1.0 then assembles this information into the format that can be imported into CogTool to automatically create an accurate UI mockup of all these webpages and links. CogTool then converts this mockup into an ACT-R device model, with which the CogTool-Explorer 1.0 model can interact.

4.1.4 Operation of CogTool-Explorer 1.0

Figure 10 illustrates an example run of the CogTool-Explorer 1.0 model in the two-column layout. Given the description of the exploration goal "Canon Law" (paragraph of text under "Item to find" at the top of the webpage) and at least one visible link on the current page (underlined text labels), CogTool-Explorer 1.0's point of visual attention starts in the top-left corner of the page (step 1) and moves its visual attention to the link "People in the United States" nearest to its current point of visual attention (step 2a), evaluates the link's infoscent with respect to the exploration goal (step 2b) and remembers the link as the best link if it has the highest infoscent so far in the page. The model may then decide to look at and evaluate another link ("Musicians and Composers", "Theology and Practices", etc, in step 2c) or it may decide to select the best link seen so far in the page (step 3). When CogTool-Explorer 1.0 decides to select the best link, it will look back at the best link "Theology and Practices", move a simulated mouse pointer over the link and click on it (step 4). In response to the click, the device model follows the link's transition to the next page, bringing the new links into the visual field of the model. Each run of the model can be different because of noise in the model, thus, the path of the model on each run is analogous to predicting the exploration choices of a single human trial.

4.1.5 Test of CogTool-Explorer 1.0

We compared the participant data from the 22 tasks in the two-column layout to predictions by CogTool-Explorer 1.0 and by AutoCWW. As explained in Section 3.1, although AutoCWW does not consider layout position, there were no models that do consider layout position available to test against, thus, we wanted to check that CogTool-Explorer 1.0 performed at least as well as AutoCWW. We first directed CogTool-Explorer 1.0 to import the actual webpages used in that experiment and automatically create the device model. To be as comparable as possible to the AutoCWW analysis, we set CogTool-Explorer 1.0 to use the same LSA values of infoscent as AutoCWW used when it evaluated the links. Because those LSA values were from -1 to +1, with links related to the goal on the order of +0.5, whereas the original SNIF-ACT 2.0's infoscent values

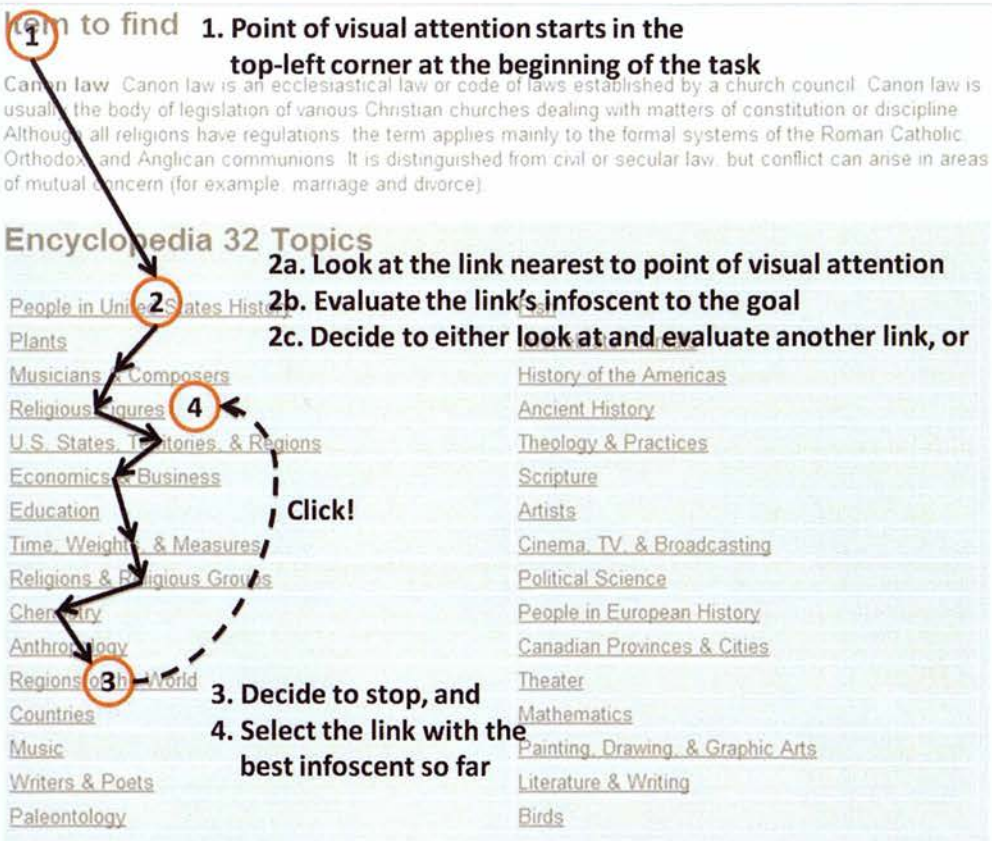


Figure 10: An example run of CogTool-Explorer 1.0 in the two-column layout.

for the same links were on the order of +25, we scaled the LSA values by a factor of 50 to be useable with CogTool-Explorer 1.0.

The model's local decision to stop and select the best link so far or continue to look at another link is not fixed, but is dependent on the infoscent of the links that have been evaluated so far, and moderated by parameters τ and k. τ is the variance of the ACT-R noise function that is applied to the infoscent value each time a link is evaluated, to reflect the variability a user might display when accessing infoscent. k (see Eq. 2) affects how rapidly the decision to stop switches as a function of the infoscent values encountered, to reflect a user's "readiness" to stop and select the best link. Fu and Pirolli (2007) explained:

> In the equation specified in the text, we set a = 1 for the read-next-link production; and a = 1 + k for the click-link production. By setting the value of a for click-link to a higher value, we assume that in general, following a link is more likely to lead to the target page than attending to the next link on the same Web page. k is a free parameter that we used to fit the data. (p. 408)

The choice of k may be partly influenced by the layout of links on the page. For example, on a query search results webpage, k would be smaller to reflect a higher readiness to stop since the most related links appear near the top of the webpage. If the layout of links is not organized, k would be larger to reflect a lower readiness to stop since the correct link might be anywhere on the webpage.

With the scaling factor of 50, we were able to use ACT-R's default noise parameter value of τ =1.0, as did Fu and Pirolli (2007). As is common practice when developing new models and tools, for example Fu and Pirolli (2007), and Halverson and Hornof (2006, 2007), we set the model's k parameter to best fit the human data. Fu and Pirolli fit their data and k was determined to be 5; the best fit to our data gives a k of 600. We believe this difference reflects the fact that Fu and Pirolli modeled search through organized webpages while our webpages present a collection of randomly organized links, therefore our model, like the participants, are more likely to keep searching on the webpage than stop and select the best link so far. With this setup, CogTool-Explorer 1.0 successfully performed the 22 tasks the same number of times as did the participants, producing predictions of *mean clicks on webpage* and *percent first click success* (Section 3.1 gives the definitions of these two performance measures).

Figure 11 shows the results by participants, CogTool-Explorer 1.0 and AutoCWW predictions on the 22 tasks in the two-column layout. A two-way ANOVA for mean clicks on webpage found that it was significantly easier to find the correct link in the left column as opposed to the right column (F (1, 60) = 8.83, $p < 0.01$), that there was a significant difference between participants and models (F (2, 60) = 89.3, $p < 0.001$), and no significant interaction (F (2, 60) = 1.65, $p = 0.20$). Post-hoc tests revealed that the participant-model effect is due to the differences between AutoCWW predictions and participant performance, and not due to CogTool-Explorer 1.0 predictions. Although the magnitudes of CogTool-Explorer 1.0 predictions were larger than participant performance, these differences between CogTool-Explorer 1.0 and participants were not significant when the correct links were in the left column (F (1, 60) = 1.47, $p = 0.23$) or in the right column (F (1, 60) = 3.59, p = 0.06). Importantly, CogTool-Explorer 1.0 captured the effect of target column observed in participants and took significantly less clicks when the correct links were in the left column than in the right column (F (1, 60) = 8.33, $p < 0.01$). In contrast, AutoCWW predicted far more clicks on the webpages than participant performance when the correct links were in the left column (F (1, 60) = 113.74, $p < 0.001$) or in the right column (F (1, 60) = 55.05, $p < 0.001$). AutoCWW also did not capture the effect of target column and did not predict any significant difference between exploration tasks with the correct link in the left or right column (F (1, 60) = 0.12, $p = 0.73$).

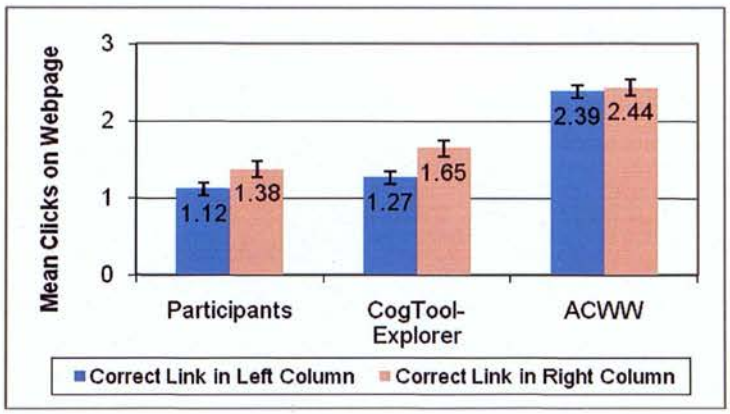


Figure 11: Mean clicks on webpage by target column, comparing participants' performance to predictions by CogTool-Explorer 1.0 and AutoCWW (Standard Error shown)

In Figure 12, two-way ANOVA for percent first click success found similar results: performance was significantly better when the correct link was in the left column than in the right column (F (1, 40) = 17.0, $p < 0.001$), the participants performed better than the model (F (1, 40) = 5.45, $p < 0.05$), and there was no significant interaction (F (1, 40) = 0.03, $p = 0.87$). Although CogTool-Explorer 1.0 predictions were less successful than participant performance, post-hoc tests reduced the significance of the differences between CogTool-Explorer 1.0 and participants when examined separately for each column (F (1, 40) = 2.88, $p = 0.10$ for the left column; F (1, 40) = 2.64, $p = 0.11$ for the right column). However, CogTool-Explorer 1.0 reliably captured the effect of target column observed in participants and had significantly higher percent first click success when the correct links were in the left column than in the right column (F (1, 40) = 9.21, $p < 0.01$). In contrast, AutoCWW does not predict percent first click success because its regression formula for mean clicks on webpage produces a minimum of 2.29 clicks per webpage; it is not derived to make meaningful first click success predictions.

These comparisons in Figures 11 and 12 show that CogTool-Explorer 1.0 predictions aligned with participant data. By accurately preserving the layout positions of links in its device model, using a serial evaluation process that will adaptively stop and select the best link so far, and a visual search strategy, CogTool-Explorer 1.0 has the necessary set of components to replicate participant behavior in the two-column layout. The device model captures enough information about the layout to enable the user model to consider layout position in its prediction. The user model, by simulating interaction one link at a time, effectively replicates the phenomenon that some links are looked at and evaluated before other links. If the model exhaustively evaluates every link, any visual search strategy and any layout position becomes inconsequential because the model will look at and evaluate all links on a webpage before making a selection. However, because the model does adaptively stop and select the best link so far, both the visual search strategy and the layout positions must be correct, or the wrong choices will be made. Therefore, replicating participant behavior and making more accurate predictions require all three components and CogTool-Explorer 1.0 uses them all successfully.

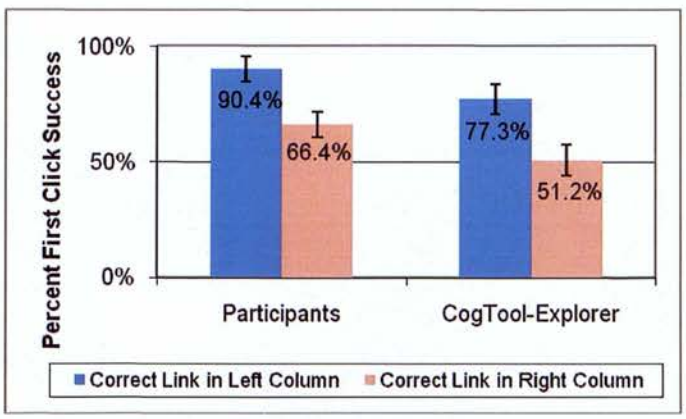


Figure 12: Percent first click success by target column, comparing participants' performance to predictions by CogTool-Explorer 1.0 (Standard error shown)

4.2 Consideration of Grouping

Section 4.1 describes the modeling work in CogTool-Explorer 1.0 to address the first research gap identified in Section 3.1: consideration of layout position. This section will describe the modeling work in CogTool-Explorer 1.1 and CogTool-Explorer 1.2, to address the second research gap identified in Section 3.2: consideration of grouping.

Miller and Remington (2004) noted that "a common approach for grouping links on a webpage involves lifting a lower level of links and placing them under each corresponding link at the upper level. In this way, two levels of the structure appear on one webpage" (p. 261). Figure 13 illustrates this with an example. Miller and Remington (2004) and Kitajima and Polson (personal communication, October 2008) suggested that users may navigate this within-webpage structure of "flatten" webpages similar to a two-level structure across multiple webpages, that is, on visiting a webpage with links laid out in groups, users will first evaluate the groups in the webpage, focus attention on a group and evaluate the links in that group. If the user decides to go back from a group, he or she will reevaluate the groups in the webpage, focus attention on another group and then evaluate the links in the new group.

Figure 14c shows an example of a webpage layout with multiple groups, which was used in the AutoCWW experiments (Blackmon et al., 2005; Toldy, 2009). The goal in this dissertation research is for CogTool-Explorer to match human performance on a layout with multiple groups like in Figure 14c. Following Miller and Remington (2004) and Kitajima and Polson (personal communication, October 2008), the approach in this dissertation research is to progressively test and modify CogTool-Explorer over three webpage layouts, starting with the *Multi-Page* layout (Figure 14a) where each link in the top-level page (also referred to as a top-level link) leads to its corresponding 2nd-level page of links, followed by the *Half-flatten* layout (Figure 14b) where

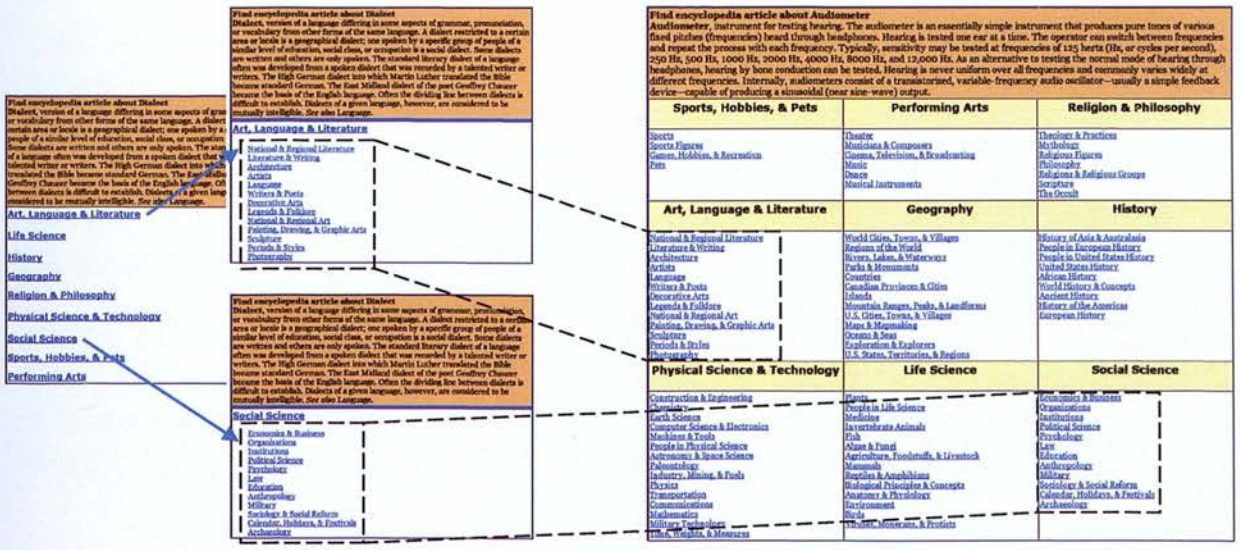
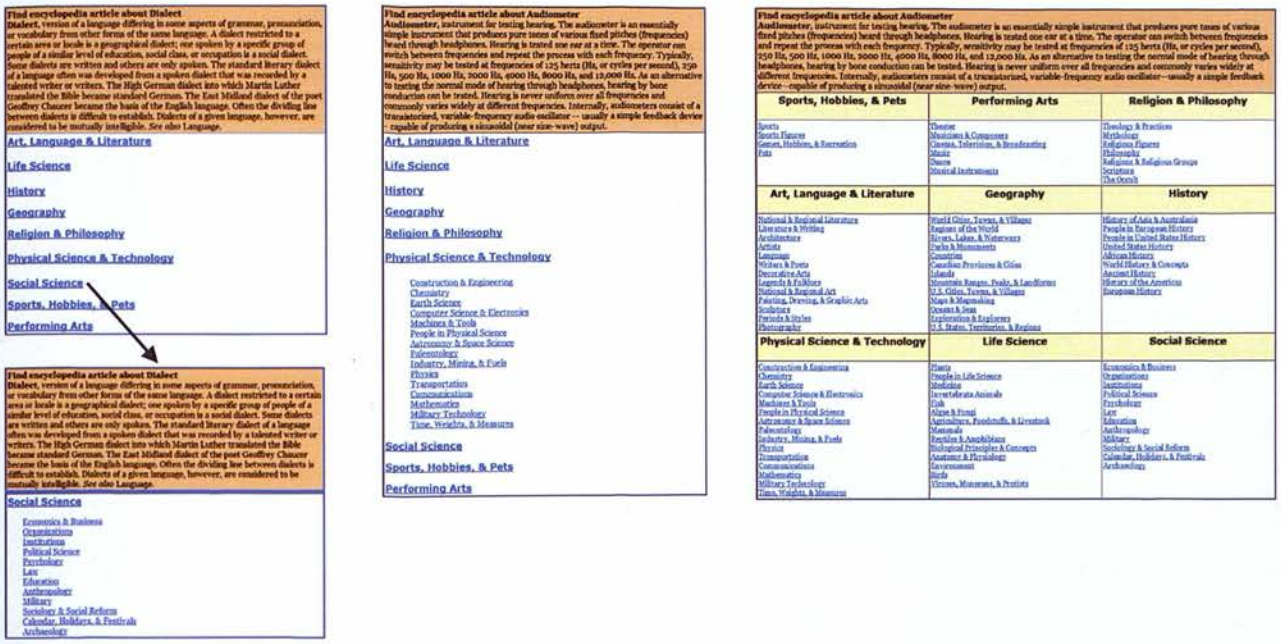


Figure 13: An example of flattening two levels of webpages (left) into a single webpage (right) by grouping the links from each 2nd-level webpage onto a single webpage. These webpage examples are from AutoCWW experiments in Blackmon et al. (2005) and Toldy (2009).



(a) (b) (c)

Figure 14: (a) The multi-page layout where each link in the top-level webpage (also referred to as a top-level link) led to its corresponding 2nd-level webpage of links, (b) the half-flatten layout where selecting a top-level link reveals the 2nd-level links grouped under that top-level link, and (c) the multi-group layout where the 2nd-level links are grouped into nine groups.

selecting a top-level link reveals the 2nd-level links grouped under that top-level link, and ending with the *Multi-Group* layout (Figure 14c) where the 2nd-level links are grouped in nine groups. These three layouts were based on an actual encyclopedia website[6] (Figure 15).

To test CogTool-Explorer, CogTool-Explorer's task performance will be compared to human data collected from participants who performed tasks on these three layouts in the AutoCWW experiments. The participant log files were generously provided by Dr. Marilyn Blackmon. The log files record each participant's sequence of link clicks and page visits.

Section 4.2.1 describes the research work done to test and modify CogTool-Explorer 1.0 in the multi-page layout, which led to CogTool-Explorer 1.1. Section 4.2.2 describes the research work done to test and modify CogTool-Explorer 1.1 in the half-flatten layout, which led to CogTool-Explorer 1.2. Section 4.2.3 describes the research work done to test CogTool-Explorer 1.2 in the multi-group layout. Section 4.2.4 compares these CogTool-Explorer models to AutoCWW's predictions in each of these three layouts. Finally, Section 4.2.5 summarizes the results and findings.

[6] http://encarta.msn.com (discontinued on October 31, 2009)

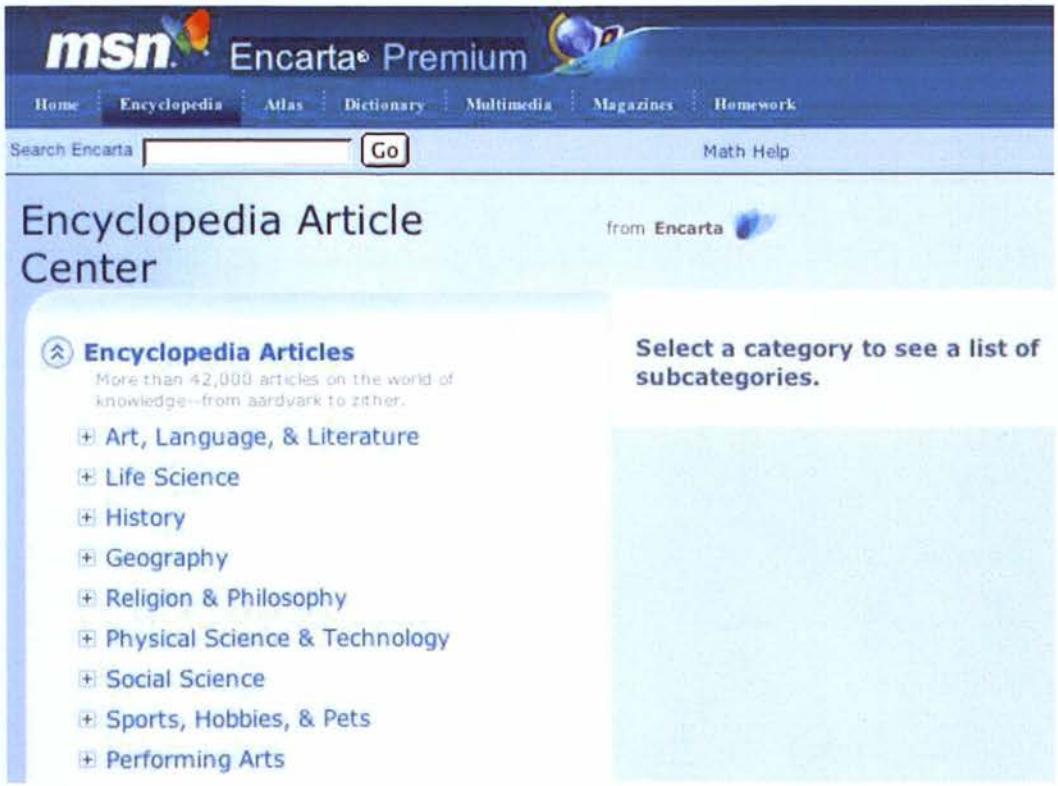


Figure 15: The Microsoft Encarta Website

4.2.1 Multi-Page Layout

The multi-page layout and tasks in this dissertation are from AutoCWW experiment Expt041022 (Experiment 1 in Toldy, 2009). In this layout (Figure 16), the task goal is presented in the paragraph of text under the line "Find encyclopedia article about..." at the top of the top-level and 2nd-level pages. Participants start in the top-level page and on selecting a link, transits to 2nd-level pages. In a 2nd-level page, participants may choose to go back to the top-level page, or select a link to go to its 3rd-level page. In a 3rd-level page, participants can check that they had succeeded in the task if the target link (for example "Dialect" in Figure 16) is in that 3rd-level page, otherwise, participants will go back to the 2nd-level page and continue exploration. Each task has only one correct 3rd-level page that contains the target link, and only one correct top-level link and only one correct 2nd-level link that lead to the correct 3rd-level page.

The multi-page layout in effect enforces a group-based hierarchical exploration process; both participants and CogTool-Explorer 1.0 have to select a "group" on the top-level page to see the links of that group in its 2nd-level page, and to select another group, both participants and CogTool-Explorer 1.0 have to go from a 2nd-level page back to the top-level page. In the multi-page layout, these group selection and go-back actions result in transitions across pages and are thus observable and recorded in the participant log files. Whereas in the half-flatten and multi-group layouts (Sections 4.2.2.2 and 4.2.3.1 respectively), most group selection and go-back actions are eye movements, thus, are not directly observable in the AutoCWW experiment setup (which had no eye-tracker) and thus not recorded in the participant log files. Therefore, this dissertation starts with the multi-page layout, which provides participant data at the necessary level of detail for comparison to model behavior, to motivate improvements to the model, before testing the improved model in the half-flatten and multi-group layouts.

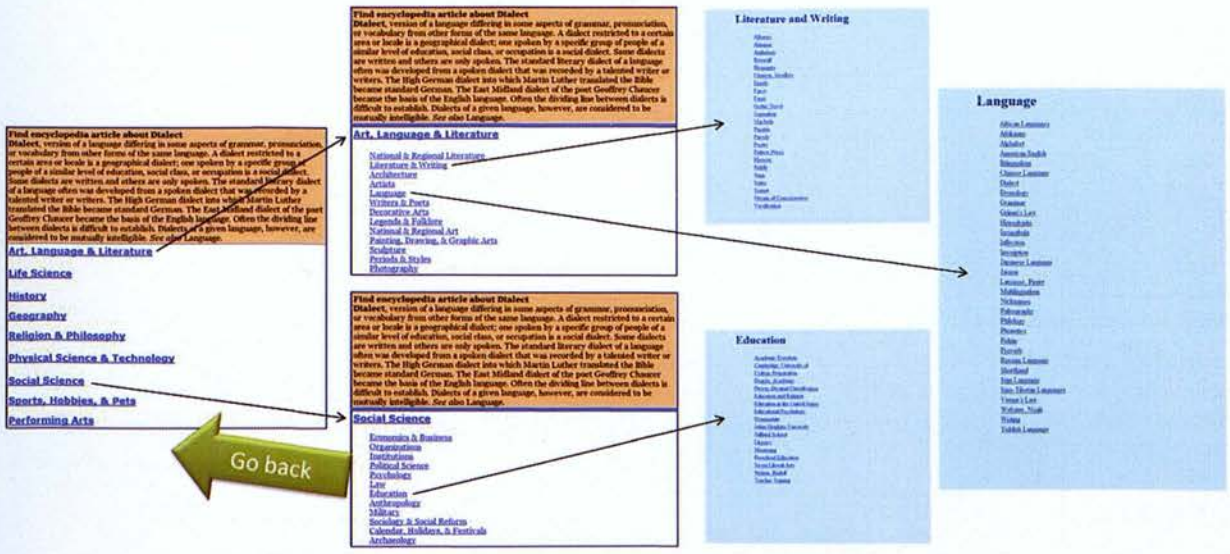


Figure 16: In the multi-page tasks, participants start on the top-level page (leftmost) and on selecting a link, transits to 2nd-level pages. Participants may go back to the top-level page, or may select a link to go to its 3rd-level page. In a 3rd-level page, participants can check if they have succeeded in the task, and if not, go back to the 2nd-level page and continue exploration.

In the multi-page layout, it is important to differentiate between going back from a 3rd level page to a 2nd-level page, versus going back from a 2nd-level page to the top-level page. Going back from a 3rd-level page is deterministic because if the target link is not in the 3rd-level page, it is clearly the wrong path and the next action will be to go back, and that is what happened in all participant trials. In contrast, going back from a 2nd-level page is a deliberate choice because the decision to go back can be taken before all 2nd-level links in the page are exhaustively selected, which is what happened in the majority of participant trials. The assumption in both the SNIF-ACT 2.0 and CogTool-Explorer 1.0 models is that the decision to go back is dependent on the infoscent of links that are looked at and evaluated, thus, it is this type of go-back actions, from a 2nd-level page to the top-level page, that is of interest in this dissertation.

Sections 4.2.1.1 and 4.2.1.2 present the operation and initial results of CogTool-Explorer 1.0 in the multi-page layout. Sections 4.2.1.3 to 4.2.1.6 present a series of refinements to the model that both increase and decrease the fit to participant data on many metrics, and eventually arrive at CogTool-Explorer 1.1 that improves on every metric compared to CogTool-Explorer 1.0. We reported the earlier parts of this series of refinements to CogTool-Explorer 1.0 (Teo and John, 2011), which is reproduced with more detail here. Section 4.2.1.7 summarizes the performance of CogTool-Explorer 1.1 and its series of refinements.

4.2.1.1 Operation of CogTool-Explorer 1.0 in the Multi-Page Layout

Figure 17 illustrates an example run of CogTool-Explorer 1.0 in the multi-page layout. Note that steps 1 to 5a in Figure 17 follows how CogTool-Explorer 1.0 operates in the two-column layout as presented in Figure 10 and Sections 4.1.1 to 4.1.4, thus this section will present steps 1 to 5a briefly. What is new in Figure 17 starts from step 5b where the model transits to a 2nd-level page that is

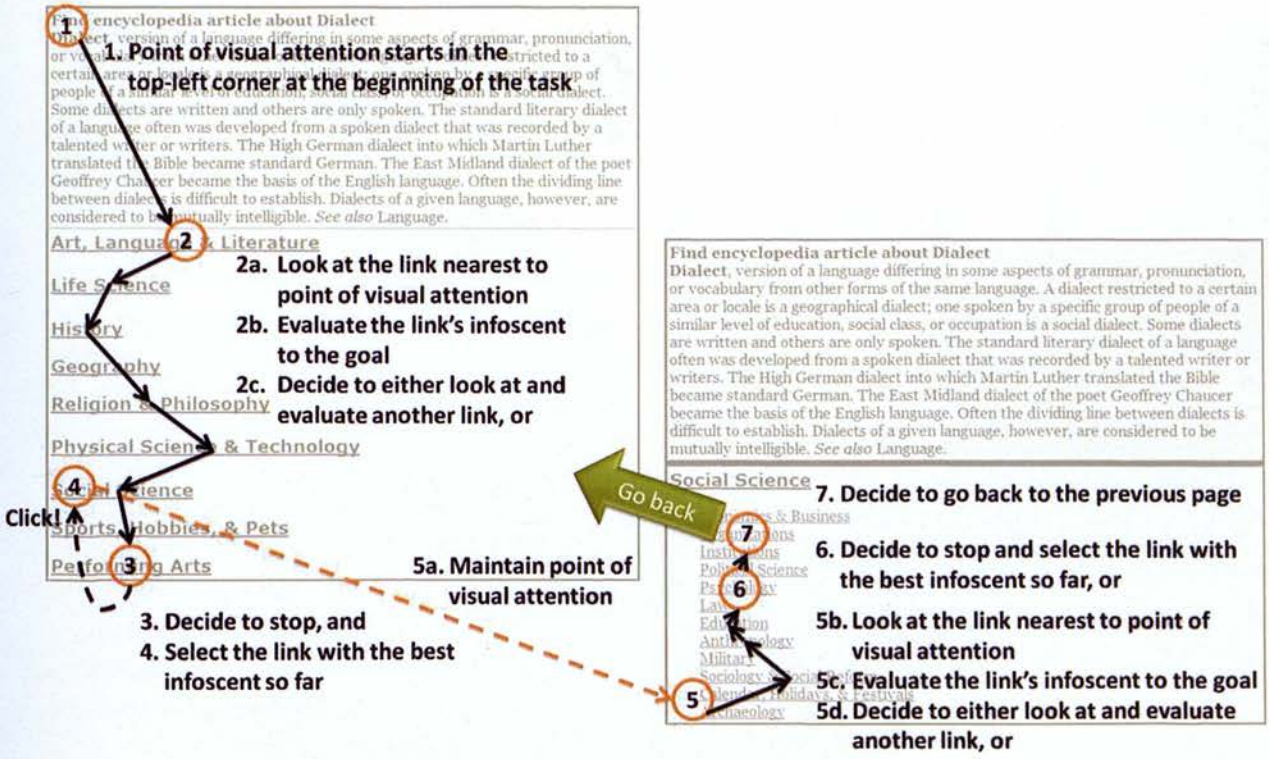


Figure 17: An example run of CogTool-Explorer 1.0 in the multi-page layout.

absent in the two-column layout.

Given the description of the task goal, CogTool-Explorer 1.0's point of visual attention starts in the top-left corner of the page (step 1), looks at the next nearest link (step 2a.), evaluates the link's infoscent with respect to the goal (step 2b) and remembers the link as the best link if it has the highest infoscent so far in the page. Base on the evaluated infoscent of the link, the model will then decide to either look at and evaluate another link (step 2c), or it may decide to stop and select the best link seen so far (step 3). When the model decides to select the best link seen so far in the page, it will look back at the best link, move a simulated mouse pointer over the link and click on it (step 4). The click causes the current page in the model's visual field to be replaced with the new page, and the model maintains its point of visual attention across page transitions (step 5a)

In the 2nd-level page, CogTool-Explorer 1.0 will proceed like it did on the previous top-level page: it will look at the nearest link (step 5b), evaluate the link's infoscent (step 5c) and remember the link as the best link if it has the highest infoscent so far in the page, and then decide to either look at and evaluate another link (step 5d), or stop and select the best link seen so far in the page (step 6), or it may decide to go back to the previous page (step 7).

This three-way decision is represented by three competing ACT-R productions and the production with the highest utility will be selected. CogTool-Explorer 1.0 uses the same utility update equations as SNIF-ACT 2.0 (see "8: Utility equations" in Fu & Pirolli, 2007) to update the utilities associated with these three productions every time after a link is evaluated. The first two actions and their utility update equations (Eq. 1 and Eq. 2) were discussed in Section 4.1.1 in the two-column layout. In a 2nd-level page of the multi-page layout, CogTool-Explorer 1.0 has the third possible action to go back to the previous page:

$$\begin{aligned} Utility_{GoBack} = \; & MIS(\text{links assessed on previous page}) \\ & - MIS(\text{links assessed on current page}) \\ & - GoBackCost \end{aligned}$$

where MIS is Mean Information Scent [Eq. 3]

Like in SNIF-ACT 2.0 (not mentioned in Fu and Pirolli, 2007, but extracted from a walkthrough of the SNIF-ACT 2.0 code), to support the computation of Eq. 3, CogTool-Explorer 1.0 uses a stack data structure to remember the infoscent of links assessed on previous pages for the first operand of Eq. 3, and a list data structure to remember the infoscent of links assessed on the current page for the second operand of Eq. 3. After a link on the current page is evaluated, its infoscent will be added to the list of infoscents for the second operand of Eq. 3. All three utility update equations will then be computed and the production with the highest utility will be selected. When the model decides to select a link and transits to a new page, the list of infoscents for the current page is pushed into the stack and the second operand is reinitialized with an empty list. When the model decides to go back to the previous page, the most recent list in the stack is removed from the stack and discarded, and the second operand is reinitialized with an empty list. Therefore, in the multi-page layout, when CogTool-Explorer 1.0 selects a 2nd-level link, transits to its 3rd-level page, find that it is not correct, and goes back to the 2nd-level page, the first operand in Eq. 3 will revert to the list of infoscents

assessed on the top-level page, and the second operand in Eq. 3 will be reinitialized with an empty list.

In CogTool-Explorer 1.0, the numeric parameter in Eq. 3, *GoBackCost*, is set to 5, the same as in SNIF-ACT 2.0 (Fu & Pirolli, 2007). If and when the model decides to go back, the model's visual field will be replaced with the previous page, and exploration continues following step 5a on the previous page.

4.2.1.2 Test of CogTool-Explorer 1.0

To test CogTool-Explorer 1.0, comparison metrics (described in the next section) are computed from model runs in the multi-group layout and from participant data in the AutoCWW experiment. In that experiment, each participant performed 36 different tasks in the multi-page layout. Each task is defined by a different task goal at the top of the page. Participants had 130 seconds to complete each task, failing which the trial is considered a failure. There were 44 to 46 valid participant trials recorded for each task.

4.2.1.2.1 Task Performance Measures and Comparison Metrics

Four task performance measures are used to measure how participants and CogTool-Explorer 1.0 performed in the tasks:

- **%Success**: the percentage of participant trials or model runs succeeding on each task. %Success is common in user testing to inform UI designers about how successful their users would be with the UI design.
- **%ErrorFreeSuccess**: the percentage of participant trials or model runs succeeding without error on each task. In the multi-page layout, this means completing the task in two clicks: selecting the correct top-level link followed by selecting the correct 2nd-level link. High %ErrorFreeSuccess indicates that the UI design for these tasks needs no improvement and therefore no further design effort. Low %ErrorFreeSuccess focuses redesign effort.
- **LinkClicks**: the number of times each link was selected in participant trials or model runs on each task. LinkClicks indicate if model runs match participant trials on choice behavior.
- **GoBacks**: the number of times participant trials or model runs went back from each 2nd-level page on each task. Indicates if model runs match participant trials on go-back behavior.

For participants, each task performance measure is calculated from participant data collected in the AutoCWW experiment. For the model, to get stable values for the above measures, multiple sets of model runs are executed until the model runs converged. To establish convergence, first, for each of the 36 tasks, 44 to 46 model runs are executed for that task, equal to the number of valid participant trials for that task, resulting in a set of model runs comprising a total of 1649 runs from all 36 tasks. From this set of model runs, *%Success* is calculated for each task. Next, an additional set of model runs is executed and combined with the previous set to form a new combined set. *%Success* is calculated for each task from this new combined set and compared to the previous set's *%Success* values. If all *%Success* values are within 1% of each other, the model runs in the combined

set are considered to have converged and no more model runs are needed. If any of the tasks in the combined set had a *%Success* value greater than 1% difference from the previous set, an additional set of model runs will be executed, combined with the previous combined set to form a new combined set and the calculated *%Success* values compared. This process is repeated until the model runs in the combined set have converged. For the models CogTool-Explorer 1.0 to CogTool-Explorer 1.1 in the multi-page layout, convergence for each model is established within 13 sets of model runs, that is, within a maximum of 21437 individual model runs over all 36 tasks. On a computer equipped with an Intel® Core™2 Duo 2.4 GHz processor and 4GB of memory, a set of 1649 model runs, where each run may run for the equivalent of 130 seconds of experiment trial time, took between half an hour to one hour to complete, depending on how successful the model was on the tasks.

After convergence, each measure is calculated from the final combined set of model runs. Since there are more model runs than participant trials for each task, *LinkClicks* and *GoBacks* by the model are normalized by the number of sets of model runs, so that these two measures are comparable to the same measures calculated from participant data. Note that *%Success* and *%ErrorFreeSuccess* do not need to be normalized since these two measures are percentages of the total number of model runs and are thus comparable to the same measures calculated from participant data.

To compare how well the model matched participant data across the 36 tasks, two comparison metrics, Correlation (expressed in R^2 value) and the Percent Average Absolute Error (%AAE, expressed as a percentage of the average observed value from participant data), are computed for each task performance measure. Correlation is a commonly used goodness-of-fit metric in the cognitive modeling literature, and the R^2 value indicates the percentage of the variance in the participant data that the model can account for. A strong correlation between model and participant data will enable the use of model predictions to differentiate between tasks at the extremes, that is, which tasks are sufficiently supported by the UI design and are mostly successful such that further design effort can be diverted to other areas, and which tasks and UI designs are less successful and thus in most need of further redesign effort. %AAE is another metric recommended by Kieras, Wood and Meyer (1997) for comparing predictive engineering models to human data. To compute %AAE, the difference between the task performance measure by participants and by the model is calculated for each task, the absolute values of these differences are then averaged and expressed as a percentage of the average task performance measure calculated from participant data. Kieras et al. (1997) argued that "the goal of engineering models is to supply predicted values of usability metrics that are not merely correlated with the empirically measured values, but are actually similar in numerical value", that "the best choice for a simple summary statistic for the accuracy of the models is the average absolute error of prediction", and "that engineers often use a rule of thumb which says that predictions accurate within 10–20% are useful for design purposes" (p. 266).

Prior work that have motivated this dissertation research had either reported correlation (for example SNIF-ACT 2.0 in Fu and Pirolli, 2007) or %AAE (for example the Minimal Model of Visual Search in Halverson and Hornof, 2007), thus, both metrics are reported in this dissertation.

4.2.1.2.2 Comparison Results

Table 2 presents the results for CogTool-Explorer 1.0 in the multi-page layout (converged after 12 sets of model runs). These results are disappointing and substantially lower than results reported by Fu and Pirolli (2007) in SNIF-ACT 2.0 for the two websites they modeled and compared to participant data. Their R^2*%Success* were 0.98 and 0.94, R^2*%LinkClicks* were 0.69 and 0.91, and R^2*%GoBacks* were 0.73 and 0.80, respectively for the two websites. Fu and Pirolli did not report R^2*%ErrorFreeSuccess.*

Table 2: CogTool-Explorer 1.0 compared to participant data in the multi-page layout

	Correlation, R^2 (95% confidence interval)	%AAE
%Success	0.28 (0.21, 0.35)	34.8%
%ErrorFreeSuccess	0.44 (0.37, 0.51)	54.2%
LinkClicks	0.25 (0.24, 0.25)	194%
GoBacks	0.25 (0.23, 0.28)	90.3%

Since CogTool-Explorer 1.0 uses the same utility update equations and model parameters values (except for parameter *k*; see Section 4.1.5) as SNIF-ACT 2.0, why are their correlation results, in particular R^2*%Success*, so different? A possible explanation is that different data collection processes are to blame. Fu and Pirolli's (2007) data are from participants doing eight tasks on each of two websites, at their leisure, on their own computers. Their participants could abandon the task at will, whereas the participant data from the AutoCWW experiment are collected in the laboratory and participants had 130 seconds to complete each task. Not compelled to continue until success, not a single participant in Fu and Pirolli's data succeeded on 4 of their 16 tasks, in contrast to the range seen in the AutoCWW experiment (average *%Success* = 71%, minimum *%Success* = 13%, maximum *%Success* = 100%). Allowing the participants to abandon tasks probably eliminated the most difficult tasks with their higher variability.

What about R^2*LinkClicks* and R^2*GoBacks*, where the results reported for SNIF-ACT 2.0 are also substantially higher than the results by CogTool-Explorer 1.0? A possible explanation is that different model-data comparison paradigms are to blame. In SNIF-ACT 2.0, these two metrics are reported from models that were run under the *model-tracing* paradigm, Fu and Pirolli explained:

> To test the predictions of the SNIF-ACT 2.0 model on its selection of links, we first started SNIF-ACT 2.0 on the same pages as the participants in all tasks. The SNIF-ACT 2.0 model was then run the same number of times as the number of participants in each task, and the selections of links were recorded. After the recordings, in case SNIF-ACT 2.0 did not pick the same Web page as participants did, we forced the model to follow the same paths as participants. This model-tracing process was a common method for comparing model predictions to human performance (e.g., see Anderson, Corbett, Koedinger, & Pelletier,

> 1995, for a review). It also allows us to directly align the model simulation results with the participant data. (p. 383)

In the model-tracing paradigm, model errors (actions that differed from what participants did) do not accumulate. When the model makes an error, the error will count against the model only once, and the model will be reset to the choice actually made in the participant trial, setting the model back on the same path as the participant trial. In contrast, for *free-running* model runs, where no realignment of model actions to participant data is performed, model errors do accumulate. Therefore, it is more difficult for a free-running model to match participant data because an error (for example a link selection on the top-level page) will take the model down a different path from the participant trial and the model may never recover and return to the path in the participant trial.

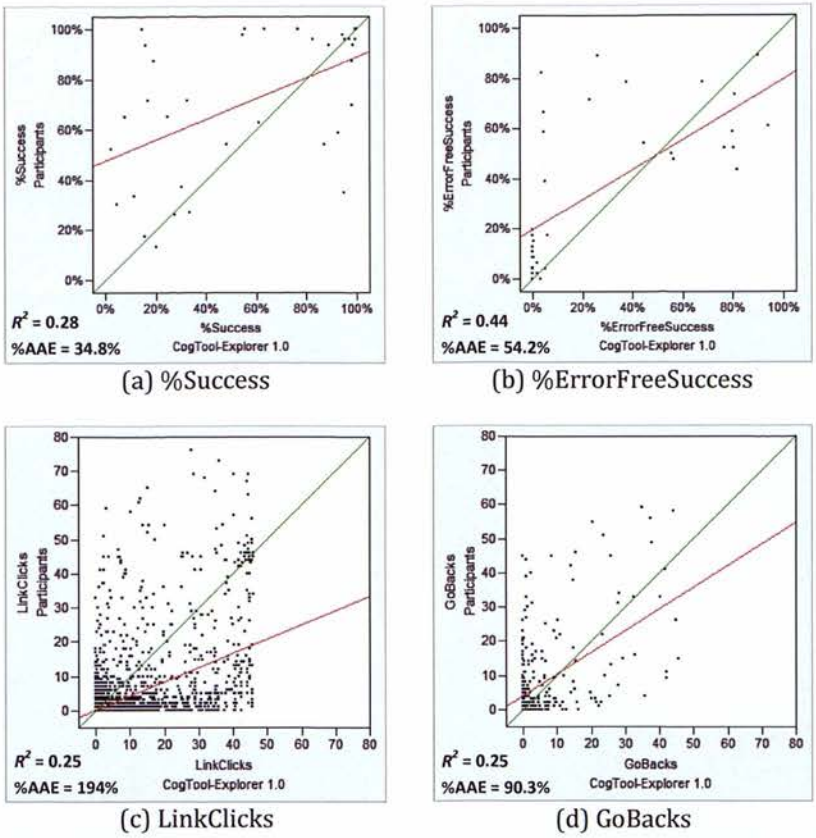
Fu and Pirroli (2007) noted that compared to results from free-running models, results from model-tracing "may not truly reflect the general capabilities of the model in predicting user–Web interactions." (p. 388). Furthermore, when HCI practitioners want *a priori* predictions of likely user exploration from a modeling tool, the predictive model has to run as a free-running model because analysis is needed before there is human data. For these reasons, results of model-data comparisons for all CogTool-Explorer models has always been from *free-running* model runs, thus, the lower R^2*%LinkClick* and R^2*%GoBacks* results by CogTool-Explorer compared to SNIF-ACT 2.0 could be due to the use of free-running models rather than the model-tracing paradigm[7].

The next 22 pages of this dissertation present a detailed explanation of the many refinements made to CogTool-Explorer 1.0 to emerge with CogTool-Explorer 1.1, which improved the fit of the model to participant data, both qualitatively and quantitatively. Some of these refinements improved fit immediately whereas others required the accumulation of several refinements to improve the fit. Readers may choose to skip to Section 4.2.1.7 for the performance of CogTool-Explorer 1.1 and its path of refinement (Figure 28).

With an understanding of the effect that free-running models have on the poor initial results of CogTool-Explorer 1.0 in the multi-page layout, the next step is to inspect at a more detailed level what might have led to the model's poor performance and what can be done to improve the match between model behavior and participant data. Figure 18 presents the scatter plots of CogTool-Explorer 1.0 compared to participant data in the multi-page layout.

---

[7] R^2*%Success* results reported by SNIF-ACT 2.0 are from free-running model runs.



(a) %Success (b) %ErrorFreeSuccess

(c) LinkClicks (d) GoBacks

Figure 18: CogTool-Explorer 1.0 compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

Two patterns of mismatch can be seen in the scatter plots of Figure 18. First, there are many tasks where participants were more successful than the model, as indicated by data points above the diagonal in Figure 18a, and to a lesser extent in Figure 18b. This suggests that on these tasks, where participants tended to select the correct top-level and 2nd-level links, the model did not. The second pattern of mismatch is evident from the blank area in the right side of Figure 18c, framed by a vertical "wall" projecting from about 45 on the horizontal axis. While there were links that participants selected more times than the number of participant trials (44 to 46 trials depending on the task), indicating that participants reselected links within a single trial, the model never selected a link more times than the number of model runs, thus, the model never reselected a link within a single run and so no data points appear beyond 46 on the horizontal axis. Sections 4.2.1.3 and 4.2.1.4 will address these two issues in turn.

4.2.1.3 Refinement of Infoscent Estimate for Top-Level Links

Figures 18a and 18b show that participants were more successful than CogTool-Explorer 1.0, thus, a good place to start is by examining the tasks where participants were most successful. Furthermore, on tasks where participants were most successful, that is, on tasks with the

highest *%ErrorFreeSuccess*, participants were least likely to be exploring in a random fashion, so a systematic model is more likely to explain the behavior. Of the tasks with the highest *%ErrorFreeSuccess* by participants, the topmost task is to search for information about "Fern". The correct top-level link is "Life Sciences" and the correct 2nd-level link is "Plants". The 46 participants only selected other top-level links 8% of the time, but went back from those incorrect 2nd-level pages to select "Life Science" and then "Plants" (in all but 2 cases) to complete the task. In contrast, CogTool-Explorer 1.0 selected other top-level links 70% of the time before selecting "Life Sciences", and on some model runs it never selected "Life Sciences" and failed the task.

One possible explanation for the model behavior is that it did not look at "Life Science" before deciding to select a link in the top-level page. When the details of the model runs were examined, this was not the case, as the model runs did see "Life Science" before selecting a link in over 95% of first-visits to the top-level page. A second possible explanation is that the model looked at too many links and saw other higher infoscent links before selecting a link in the top-level page. This also was not the case because in all model runs up to the point where it finished looking at "Life Science", if the model was forced to choose the best link so far, it would have selected "Life Science" in over 60% of the model runs. A third possible explanation lies in the infoscent values used by the model.

Given a particular task goal, CogTool-Explorer 1.0 uses LSA to compute an infoscent value for each link, based on the cosine value between two vectors, one representing the words in the task goal and the other the words in the link text, which is the same method used by AutoCWW (Blackmon, et al., 2005). To approximate how a reader elaborates and comprehends the link text in relation to his or her background knowledge, AutoCWW identifies all the words in the selected LSA corpus that occur at least 50 times in the corpus and have a minimum cosine of 0.5 with the link text vector, and appends these words to the link text as an elaboration, before using LSA to compute the infoscent of the link. Kitajima et al. (2005) explained that "elaborated link labels generally produce more accurate estimates of semantic similarity (LSA cosine values)." CogTool-Explorer 1.0 used this same elaboration method, thus, for the link "Life Science", the words "science sciences biology scientific geology physics life biologist physicists" are appended to compute the link's infoscent value.

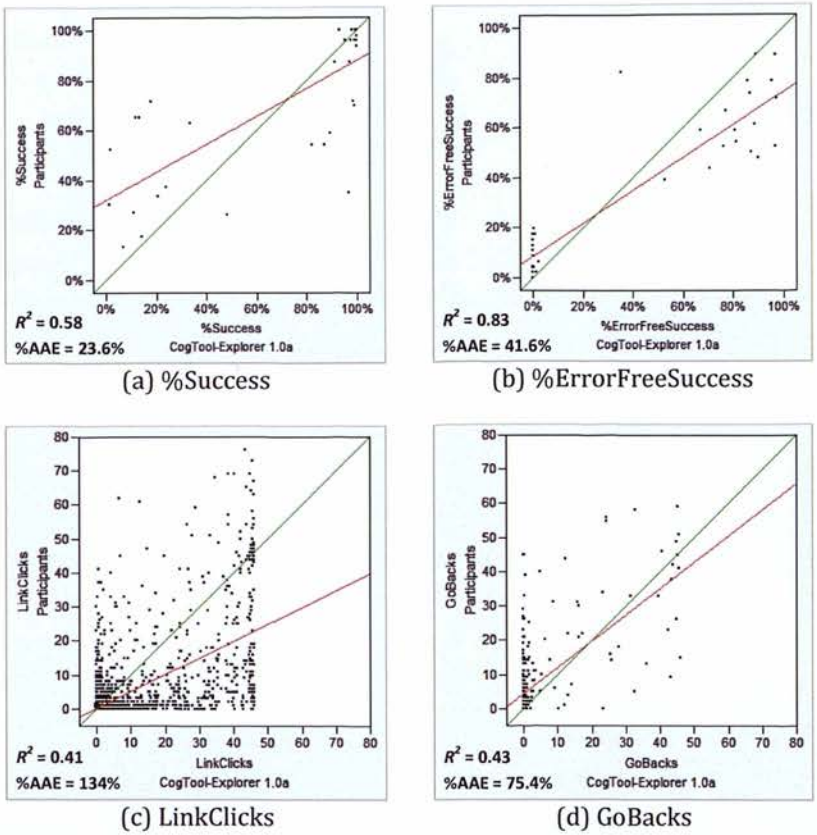
In a multi-group layout (Figure 14c), where links are grouped into regions labeled with a heading, AutoCWW uses a further elaboration method. Kitajima et al. (2005) explained that "readers scan headings and subheadings to grasp the top-level organization or general structure of the text". To represent a region, AutoCWW first elaborates (as described in the previous paragraph) the heading text and the link text of each link of that region, and then appends the elaborated link texts to the elaborated heading text to form a text string from which to compute the region's infoscent. CogTool-Explorer 1.0 did not use this elaboration method for top-level links in the multi-page layout because their subordinate links appeared on 2nd-level pages, different from Kitajima et al.'s assumption. However, participants did practice trials on this same multi-page layout as the actual test trials, and performed all 36 test trials on this same multi-page layout. Therefore, it is a reasonable assumption that the information gained from this experience would influence how participants assessed the infoscent of a top-level link. This reasoning motivated the refinement in CogTool-Explorer 1.0a to better represent these participants: for the infoscent of a top-level link, the top-level link will be elaborated (as described in the previous paragraph) and then appended

with the link texts from its associated 2nd-level links. While this refinement is similar to AutoCWW's procedure, the justifications are different.

Table 3 presents the results for CogTool-Explorer 1.0a in the multi-page layout (converged after 12 sets of model runs). CogTool-Explorer 1.0a improved over CogTool-Explorer 1.0 on all eight metrics; *R²%Success* more than doubled and *R²%ErrorFreeSuccess* almost doubled. Figure 19 shows the scatter plots of CogTool-Explorer 1.0a compared to participant data. Comparing Figures 19a to 18a on *%Successes*, there is a movement of those data points above the diagonal (tasks on which participants were successful) towards the right (CogTool-Explorer 1.0a have became as successful). This improvement can also been seen in *%ErrorFreeSuccesses* by comparing Figures 19b to 18b. There appear to be at least two outlier tasks, one in the lower right quadrant of Figure 19a and the other in the upper left quadrant in Figure 19b. As data points move over the course of further model refinements through Sections 4.2.1.4 to 4.2.1.6.2, outlier tasks that remain will be discussed in Section 4.2.1.6.2 in the test of CogTool-Explorer 1.1.

Table 3: CogTool-Explorer 1.0 and 1.0a compared to participant data in the multi-page layout. The better results are highlighted in bold.

	Correlation, R^2 (95% confidence interval)		%AAE	
	CogTool-Explorer **1.0**	CogTool-Explorer **1.0a**	CogTool-Explorer **1.0**	CogTool-Explorer **1.0a**
%Success	0.28 (0.21, 0.35)	**0.58** (0.52, 0.64)	34.8%	**23.6%**
%ErrorFreeSuccess	0.44 (0.37, 0.51)	**0.83** (0.79, 0.85)	54.2%	**41.6%**
LinkClicks	0.25 (0.24, 0.25)	**0.41** (0.40, 0.41)	194%	**134%**
GoBacks	0.25 (0.23, 0.28)	**0.43** (0.40, 0.45)	90.3%	**75.4%**



(a) %Success (b) %ErrorFreeSuccess

(c) LinkClicks (d) GoBacks

Figure 19: CogTool-Explorer 1.0a compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

An argument against the refinement introduced in CogTool-Explorer 1.0a is that it makes 2nd-level links influence the infoscent of a top-level link before the model visits the 2nd-level page. It is like helping the model to select the correct top-level link by "peeking" at where the correct 2nd-level link is located. However, this refinement can also hurt the model. For example, one of the tasks is to find the page for the task goal "Dance of Death", where participants' *%Success* was 63%, but this refinement increases the infoscent of competing top-level links, and *%Success* by CogTool-Explorer 1.0 dropped from 61% to 34% in CogTool-Explorer 1.0a.

A more general argument for introducing this refinement in CogTool-Explorer 1.0a and to bring this refinement forward as we continue to improve CogTool-Explorer is that this refinement has a similar effect as Budiu and Pirolli's (2007) use of category-based scent in DOI-ACT as described in Section 2.3.6. Budiu and Pirolli argued that "a more appropriate measure for information organized taxonomically is degree of category membership: how much the target is a member of the class denoted by the label". In DOI-ACT, Budiu and Pirolli had "self-selected participants rated, on a scale from 1 to 5, how likely it is for a node (e.g., banana) to be member of a class (e.g., fruits), and these category ratings were used by DOI-ACT to guide its selections during exploration". In the multi-

page layout, the top-level and 2nd-level pages are organized taxonomically, thus, this refinement in CogTool-Explorer 1.0a has a similar effect of adding category cues to the top-level links, to improve the quality of infoscent estimates for these links. In future work, using online sources of taxonomic information, such as WordNet (Miller, Beackwith, Fellbaum, Gross, & Miller, 1990), instead of the specially-collected human categorizations in DOI-ACT, may be an interesting addition, but in this dissertation we will continue with this refinement in CogTool-Explorer 1.0a.

While this refinement in CogTool-Explorer 1.0a resulted in improvements, the second pattern of mismatch described in Section 4.2.1.2 remained, where participants did reselect links within a single trial but the model never did, which is evident from the blank area on the right side of the vertical "wall" in Figures 18c and 19c. The next section will address this mismatch.

4.2.1.4 Addition of Reselection Behavior

Participant data reveal that about 13% of all link selections by participants were reselections of links that they had previously selected in that same trial, but CogTool-Explorer 1.0a never reselected previously selected links. This model behavior is shared with both CogTool-Explorer 1.0 and SNIF-ACT 2.0. When these models select a link, the ACT-R visual object that represents the link will be marked as having been selected, and that visual object will be added to the ACT-R declarative memory, so that the model can retrieve it from declarative memory perfectly, check that the link is marked as selected and will not select it again (not reported in Fu and Pirolli, 2007, but extracted from a walkthrough of the SNIF-ACT 2.0 code). Presumably, since Fu and Pirolli's data come from tasks performed by participants in naturalistic settings using web browsers on their own personal computers, the link color changed when a link had been previously selected and thus this "perfect memory" was "in the world". This mechanism remained unchanged in CogTool-Explorer 1.0 and 1.0a.

However, participants did reselect links that they had previously selected in that same trial. This means that the mechanism in CogTool-Explorer 1.0a that perfectly remembered which links had been clicked on and never reselect them must be changed to allow the possibility of matching participant behavior. It is not possible to tell from the participant data whether a reselection was a deliberate decision to select the link a second time or that the participant forgot that link had been previously selected (in the AutoCWW experiment where the participant data was collected, the web browser was set up to not change the color of links when links were selected); in the absence of empirical evidence, I decided to model the latter in CogTool-Explorer 1.0b using ACT-R's activation decay mechanism as explained in the next paragraph.

Each link is represented as a visual object (an ACT-R chunk) that has a *status* attribute (an ACT-R chunk slot) whose value is set to *chosen* when the link is selected by the model and then stored in declarative memory. Instead of being able to retrieve this piece of information perfectly like in CogTool-Explorer 1.0a, CogTool-Explorer 1.0b utilizes ACT-R's chunk activation decay mechanism to govern whether the fact that the link has been previously selected will be successfully retrieved from declarative memory when this link is next looked at and evaluated by the model. Four ACT-R parameters govern this decay mechanism. Three of these parameters, base-level learning (:bll), permanent noise (:pas) and instantaneous noise (:ans), have recommended or commonly used values (:bll = 0.5, :pas = nil and :ans = 1) and are set to these values in CogTool-Explorer 1.0b. The

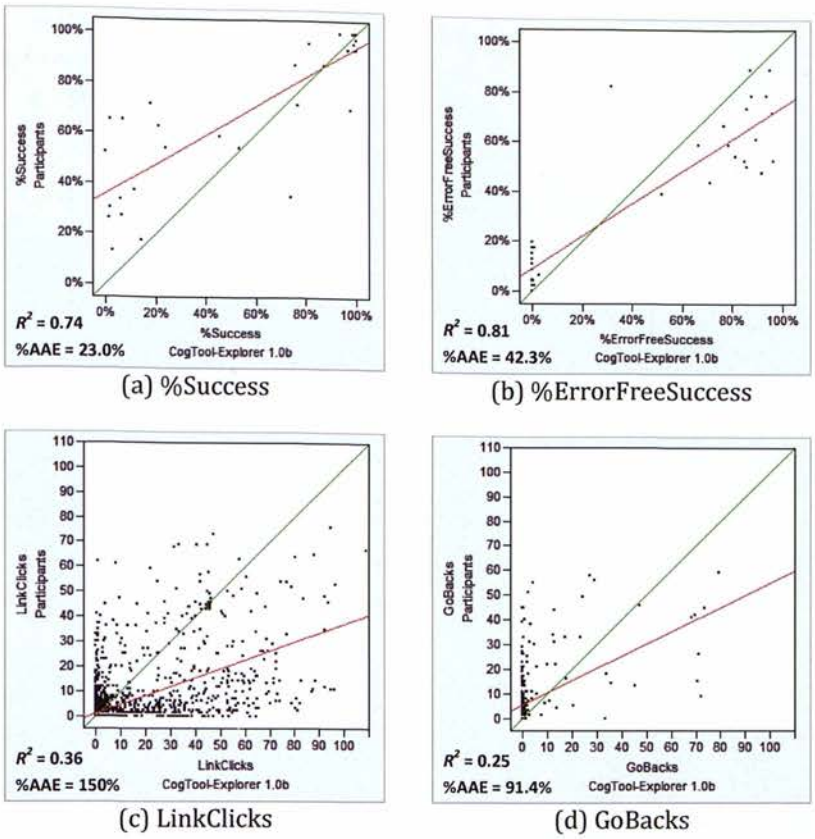
fourth parameter, activation retrieval threshold (:rt), determines if the model makes a retrieval request and there is a matching chunk, that is, the link was previously selected, that chunk will only be retrieved if it's activation exceeds the retrieval threshold. In CogTool-Explorer 1.0b, :rt is set to -2 to approximate a 50% chance of reselection midway through the 130s task duration in the experiment.

Table 4 presents the results for CogTool-Explorer 1.0b in the multi-page layout (converged after 8 sets of model runs). Compared to CogTool-Explorer 1.0a, CogTool-Explorer 1.0b improves on the match to participant data in *%Success*, and is comparatively unchanged in *%ErrorFreeSuccess*. However, CogTool-Explorer 1.0b has a poorer match to participant data on *LinkClicks* and *GoBacks* compared to CogTool-Explorer 1.0a.

Table 4: CogTool-Explorer 1.0a and 1.0b compared to participant data in the multi-page layout. The better results are highlighted in bold.

	Correlation, R^2 (95% confidence interval)		%AAE	
	CogTool-Explorer **1.0a**	CogTool-Explorer **1.0b**	CogTool-Explorer **1.0a**	CogTool-Explorer **1.0b**
%Success	0.58 (0.52, 0.64)	**0.74** (0.68, 0.78)	23.6%	**23.0%**
%ErrorFreeSuccess	**0.83** (0.79, 0.85)	0.81 (0.77, 0.85)	**41.6%**	42.3%
LinkClicks	**0.41** (0.40, 0.41)	0.36 (0.35, 0.37)	**134%**	150%
GoBacks	**0.43** (0.40, 0.45)	0.25 (0.22, 0.28)	**75.4%**	91.4%

Comparing Figures 20c to 19c, enabling reselection of previously selected links in CogTool-Explorer 1.0b has the desired effect: in Figure 20c, some links that were reselected by participants (data points with y-coordinates greater than 46) were also reselected by CogTool-Explorer 1.0b (data points with x-coordinates greater than 46). This is a marked difference compared to the blank area on the right side of the vertical "wall" in Figures 19c and 18c.



(a) %Success (b) %ErrorFreeSuccess

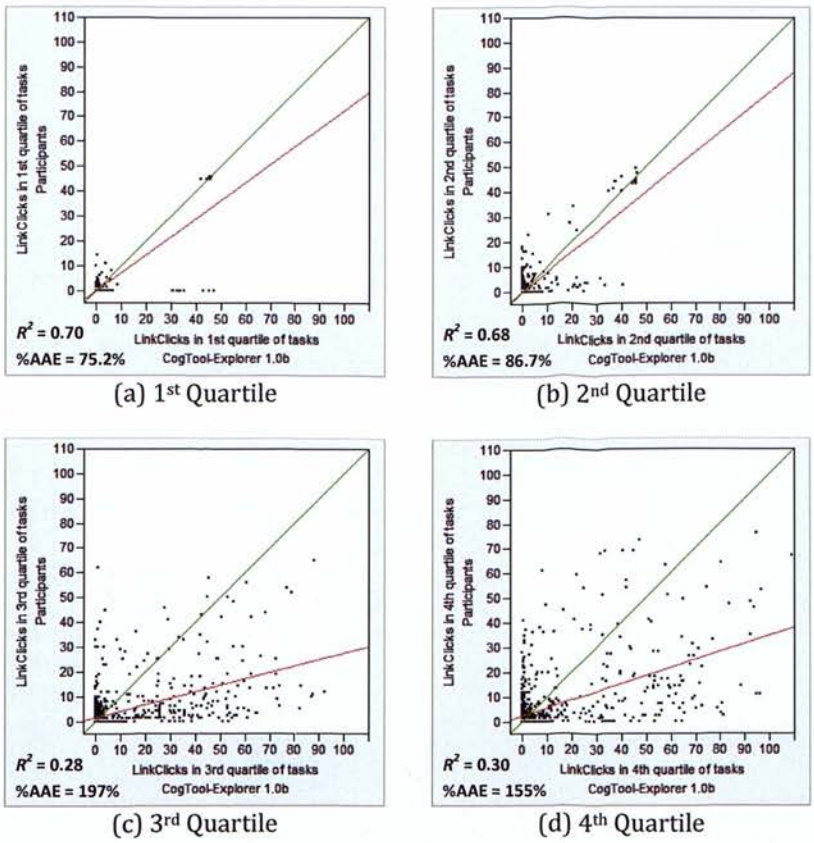
(c) LinkClicks (d) GoBacks

Figure 20: CogTool-Explorer 1.0b compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

However, the preponderance of data points to the right of the green diagonal in Figure 20c indicates that CogTool-Explorer 1.0b was overly aggressive in reselecting some links. To investigate this further, the 36 tasks are sorted by decreasing *%Success* by participants and grouped into 4 quartiles of 9 tasks each. Table 5 breaks down the *LinkClicks* metric by these 4 quartiles and Figure 21 shows the corresponding scatter plots for *LinkClicks* in these 4 quartiles (Figure 21 breakouts from Figure 20c by these 4 quartiles).

Table 5: The 36 tasks in the multi-page layout sorted by decreasing %Success by participants and grouped into 4 quartiles of 9 tasks each. Table also shows the breakdown of %Success and LinkClicks by CogTool-Explorer 1.0b in these 4 quartiles.

		Quartile of tasks (sorted by descending %Success by participants)			
		1st	2nd	3rd	4th
%Success Mean (Max, Min)	Participants	99.5% (100%, 97.8%)	93.0% (95.7%, 87.0%)	63.8% (71.7, 54.4%)	30.2% (52.2%, 13.2%)
	CogTool-Explorer 1.0b	98.6% (100%, 93.3%)	92.9% (99.7%, 75.8%)	38.3% (97.6%, 1.9%)	13.2% (73.6%, 0.3%)
LinkClicks by CogTool-Explorer 1.0b	Correlation, R^2 (95% confidence interval)	0.70 (0.69, 0.71)	0.68 (0.68, 0.70)	0.28 (0.26, 0.29)	0.30 (0.28, 0.32)
	%AAE	75.2%	86.7%	197%	155%



(a) 1st Quartile (b) 2nd Quartile

(c) 3rd Quartile (d) 4th Quartile

Figure 21: LinkClicks by CogTool-Explorer 1.0b compared to participant data, broken down by the four quartiles shown in Table 5. Each data point represents a link in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

For the tasks in the 1st and 2nd quartiles, participants were very successful, averaging 99.5% and 93.0% success respectively (first row of Table 5), and CogTool-Explorer 1.0b was also almost as successful, averaging 98.6% and 92.9% success respectively (second row of Table 5). For these two quartiles of tasks, CogTool-Explorer 1.0b matched participants on LinkClicks well, achieving *R²LinkClicks* of 0.70 and 0.68 respectively (third row of Table 5), which are higher than the *R²LinkClicks* of 0.36 when computed from all 36 tasks (third row of Table 4), and are comparable to the *R²LinkClicks* of 0.69 and 0.91 reported for SNIF-ACT 2.0 on two websites modeled by Fu and Pirolli (2007). As discussed in Section 4.2.1.2.2, *R²LinkClicks* reported for SNIF-ACT 2.0 are from model-tracing runs, whereas the *R²LinkClicks* by CogTool-Explorer 1.0b are from free-running models, which make it more difficult for CogTool-Explorer 1.0b to match participant data.

For the tasks in the 3rd and 4th quartiles, participants were less successful, averaging 63.8% and 30.2% success respectively (first row of Table 5), and CogTool-Explorer 1.0b was even less successful, averaging 38.3% and 13.2% success respectively (second row of Table 5). The lower *%Success* by CogTool-Explorer 1.0b compared to participants meant that there were substantially more model runs than participant trials that continued to select links until the task time of 130 seconds was up. This would explain why in contrast to the 1st and 2nd quartiles of tasks (Figures 21a and 21b), there were more link selections and reselections by CogTool-Explorer 1.0b compared to participants in the 3rd and 4th quartiles, as evident from the preponderance of data points to the right of the diagonal in Figures 21c and 21d. As a free-running model, CogTool-Explorer 1.0b could make more link selections than participants did, whereas in the model-tracing approach used by SNIF-ACT 2.0, the model would make exactly the same number of link selections as participants did. When all 36 tasks from all four quartiles are put together in Figure 20c, the free-running model runs in tasks where CogTool-Explorer 1.0b was less successful than participants could account for the overly aggressive selection and reselection of links by the model. We will return to this issue again in Section 4.2.1.6.

In summary, this refinement in CogTool-Explorer 1.0b has the expected effect on model behavior, and resulted in an improvement in *%Success* and maintained *%ErrorFreeSuccess*. While the refinement resulted in worse results for *LinkClicks*, the detailed breakdown and analysis in the preceding two paragraphs could provide an account for it. This refinement also resulted in worse results for *GoBacks*; the next section will investigate the model's go-back behavior.

4.2.1.5 Refinement of GoBack Behavior

On visiting a 2nd-level page in the multi-page layout, participants would take one of two possible actions: select a link or go back to the top-level page. Participant data revealed that 26% of participant actions in 2nd-level pages were go-backs, but only 6% of CogTool-Explorer 1.0b actions in 2nd-level pages were go-backs. Interestingly, in the tasks Fu and Pirolli's (2007) used for SNIF-ACT 2.0, 5% of their participant actions were go-backs. This difference in the percentage of go-back actions by participants (26% in the multi-page tasks versus 5% in the SNIF-ACT 2.0 tasks) may be because of the different data collection processes used. As explained in Section 4.2.1.2.2, participant data for the multi-page tasks were collected in a laboratory setting and participants had 130 seconds to complete each task. In contrast, participants in Fu and Pirolli's (2007) data performed their tasks at their leisure, on their own computers, and could abandon tasks at will. Allowing the

participants to abandon tasks instead of going to completion probably shortened or even eliminated the most difficult tasks where participants would likely have to make more go-back actions to succeed in the task.

As explained in Section 4.2.1.1, on visiting a 2nd-level page, after looking at and assessing the infoscent of a link, CogTool-Explorer 1.0b will choose between reading another link, selecting the best link seen so far, or going back to the previous page. This three-way decision is represented by three competing ACT-R productions and the production with the highest utility will be selected. CogTool-Explorer 1.0b uses the same utility update equations as SNIF-ACT 2.0 (see 8: Utility equations in Fu & Pirolli, 2007) to update the utilities associated with the above three actions each time after a link is evaluated. The utility update equations for reading another link and selecting the best link so far have both strong theoretical support (Fu & Pirolli, 2007) and empirical support from several studies that did not emphasize or use go-back behavior (Fu & Pirolli, 2007, and Teo & John, 2008). Compared to those two utility update equations, the utility update equation for going back has relatively less theoretical and empirical support:

$$\begin{aligned} \text{Utility}_{\text{GoBack}} = \quad & \text{MIS(links assessed on previous page)} \\ & - \text{MIS(links assessed on current page)} \\ & - \text{GoBackCost} \end{aligned}$$

where MIS is Mean Information Scent [Eq. 3]

CogTool-Explorer 1.0b uses this same utility update equation (Eq. 3) and the same GoBackCost parameter value of 5 as SNIF-ACT 2.0 to govern when the model chooses to go back to the previous page. This may be a reason why CogTool-Explorer 1.0b had 6% of go-back actions, which is comparable to the 5% of go-back actions that Fu and Pirolli's (2007) participants did in their tasks. However, this is substantially less than the 26% of go-back actions by participants in the multi-page tasks, thus, it calls into question the utility equation and the GoBackCost parameter that govern CogTool-Explorer 1.0b's go-back behavior.

The only description Fu and Pirolli (2007) gave for GoBackCost was "the cost of going back to the previous page" (p. 381), and in SNIF-ACT 2.0, its value was set to 5, presumably providing a good fit to their data. Table 6 shows how the percentage of go-back actions from 2nd-level pages by CogTool-Explorer 1.0b changed as a function of the GoBackCost parameter. At the GoBackCost value of 1, the percentage of go-back actions by the model increased to 24%, almost on par with the 26% by participants. A GoBackCost parameter value of zero does not make sense because the parameter represents the cost of navigating to the previous page on a web browser versus remaining on the current page, and there is at least some cost in time and effort to perform that navigation. Thus, CogTool-Explorer 1.0c uses the GoBackCost parameter value of 1.

Table 6: Percentage of go-back actions from 2nd-level pages by CogTool-Explorer 1.0b as a function of the GoBackCost parameter, compared to participants. CogTool-Explorer 1.0c uses the GoBackCost parameter value of 1.

	CogTool-Explorer 1.0b				CogTool-Explorer 1.0c	Participants
GoBackCost	5	4	3	2	1	
Percentage of go-back actions	6%	7%	9%	14%	24%	26%

Table 7 presents the results for CogTool-Explorer 1.0c in the multi-page layout (converged after 12 sets of model runs). While the reduction in the GoBackCost parameter value to 1 brought the percentage of go-back actions by the model to almost on par with that by participants (Table 6), it resulted in worse results for almost all the metrics in Table 7, except for an increase in R^2*GoBacks*. Of the correlation metrics, R^2*%Success* by CogTool-Explorer 1.0c is reduced to a third of that by CogTool-Explorer 1.0b. Comparing Figures 22a to 20a, and 22b to 20b, there is a reappearance of tasks where participants were more successful than the model, as indicated by the data points above the diagonal in Figures 22a and 22b, reminiscent of Figures 18a and 18b. The next section will examine these tasks.

Table 7: CogTool-Explorer 1.0b and 1.0c compared to participant data in the multi-page layout. The better results are highlighted in bold.

	Correlation, R^2 (95% confidence interval)		%AAE	
	CogTool-Explorer **1.0b**	CogTool-Explorer **1.0c**	CogTool-Explorer **1.0b**	CogTool-Explorer **1.0c**
%Success	**0.74** (0.68, 0.78)	0.25 (0.18, 0.32)	**23.0%**	40.4%
%ErrorFreeSuccess	**0.81** (0.77, 0.85)	0.59 (0.52, 0.64)	**42.3%**	47.5%
LinkClicks	**0.36** (0.35, 0.37)	0.35 (0.35, 0.36)	**150%**	187%
GoBacks	0.25 (0.22, 0.28)	**0.31** (0.29, 0.33)	**91.4%**	175%

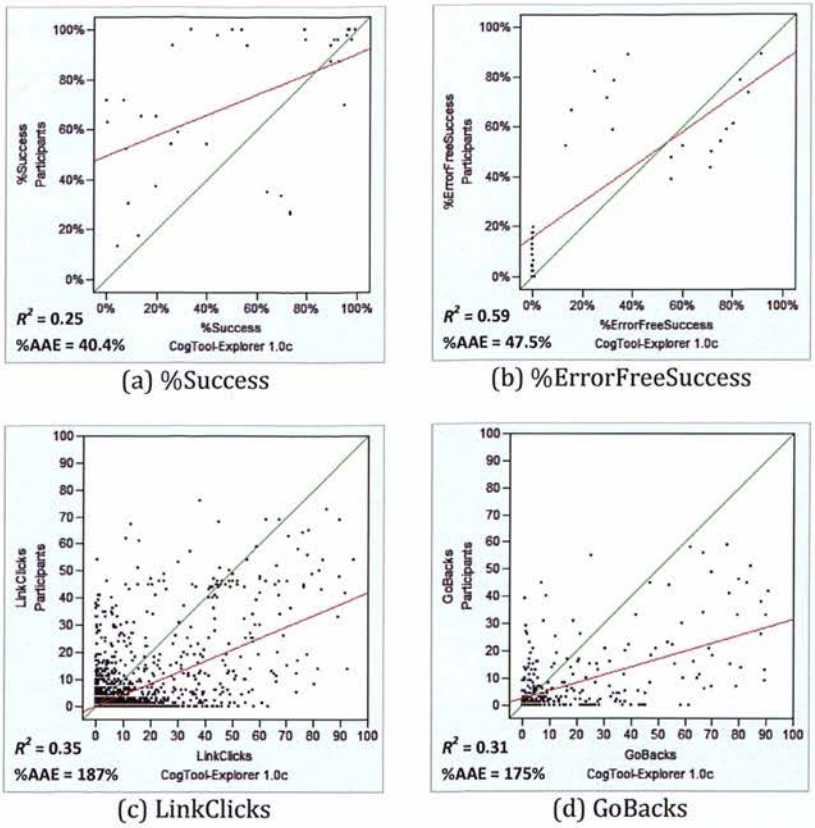


Figure 22: CogTool-Explorer 1.0c compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

4.2.1.5.1 Addition of Confidence

In CogTool-Explorer 1.0c, there is a reappearance of tasks where participants were more successful than the model. To investigate, we examine the tasks where participants were most successful, that is, on task with the highest *%ErrorFreeSuccess*, where participants would be expected to be least likely to be exploring in a random fashion, and where a systematic model will have a better chance of explaining the behavior. After sorting the 36 tasks by highest *%ErrorFreeSuccess* by participants, the topmost tasks are Fern, Niagara River, Pigeon and Hubble Space Telescope. In the Fern task, CogTool-Explorer 1.0c was only 4% less successful than participants, a margin of error acceptable for an engineering model and unlikely to reduce unless we over-fit the data. In the Pigeon task, although CogTool-Explorer 1.0c was 19% less successful than participants, poorly estimated infoscent values could explain for the under-performance by the model and this will be discussed in Section 4.2.1.6.2. Therefore, this investigation focuses on the Niagara River and the Hubble Space Telescope tasks, where CogTool-Explorer 1.0c was less successful than participants by 47% and 49% respectively.

In the Niagara River task, inspection of the CogTool-Explorer 1.0c model runs and participant trials in more detail reveal that the model was going back from 2nd-level pages when participants did not.

Participants never went back from the 2nd-level Geography page after selecting the correct top-level Geography link. However, CogTool-Explorer 1.0c went back from the 2nd-level Geography page 60% of the time after selecting the correct top-level Geography link. This suggests that the links in the top-level page looked very attractive to CogTool-Explorer 1.0c and were pulling the model back to the top-level page. For the Niagara River task, the infoscent of the correct top-level Geography link (LSA cosine value = 0.235) is an order of magnitude higher than the other top-level link with the next highest infoscent (top-level History link with LSA cosine value = 0.049). In fact, the infoscent of all the other top-level links other than Geography and History had negative LSA cosine values. So why was the model going back from the 2nd-level Geography page when the alternative links on the top-level page were not that attractive? Let's revisit the GoBack utility update equation:

$$\begin{aligned} \text{Utility}_{\text{GoBack}} = \; & \text{MIS(links assessed on previous page)} \\ & - \text{MIS(links assessed on current page)} \\ & - \text{GoBackCost} \end{aligned}$$

where MIS is Mean Information Scent [Eq. 3]

Consider a model run where CogTool-Explorer 1.0c evaluated and decided to choose the top-level Geography link. After CogTool-Explorer 1.0c selected the top-level link with the highest assessed infoscent and transited to its 2nd-level page, Eq. 3 included the high infoscent value of the Geography link in its first operand and attracted the model back to the top-level page! This behavior violates common sense; since the model had just selected the best top-level link to visit its 2nd-level page, it should not be pulled back to the previous page by the infoscent of that selected top-level link. This reasoning motivated a change to the GoBack utility update equation, to exclude the selected link from the previous page when computing the mean infoscent of the previous page, thus, changing Eq. 3 to Eq. 4:

$$\begin{aligned} \text{Utility}_{\text{GoBack}} = \; & \text{MIS(links assessed on previous page } \mathbf{\text{excluding the selected link}}\text{)} \\ & - \text{MIS(links assessed on current page)} \\ & - \text{GoBackCost} \end{aligned}$$

where MIS is Mean Information Scent [Eq. 4]

In the Hubble Space Telescope task, inspection of the CogTool-Explorer 1.0c model runs and participant trials in more detail also reveal that the model was going back from 2nd-level pages when participants did not. Participants never went back from the 2nd-level *Physical Science and Technology* page after selecting the correct top-level *Physical Science and Technology* link. However CogTool-Explorer 1.0c went back from the 2nd-level *Physical Science and Technology* page 66% of the time after selecting the correct top-level *Physical Science and Technology* link. Further investigation reveal a different problem from that in the Niagara River task, however. As explained in section 4.2.1.1, on transiting to a new page, the second operand in Eq. 3 is reinitialized to an empty list. After CogTool-Explorer 1.0c selected the best link it saw on the top-level page and transited to its 2nd-level page, if the first link the model saw on the 2nd-level page had very low infoscent, the value of the second operand in the Goback utility update equation, that is, the mean information scent of links assessed on current page, would be low, and that would make the GoBack utility high. In the Hubble Space Telescope task, the GoBack utility could be high enough for the

model to choose to go back after evaluating just the first link in the 2nd-level *Physical Science and Technology* page, when that link happened to be a low infoscent link.

However, this behavior also violates common sense; since the model had just selected the best link in the top-level page because it looked most promising. The model should carry that confidence into the next page and should not immediately go back just because the first link it saw on the 2nd-level page did not relate to the task goal. This reasoning motivated another change to the GoBack utility update equation, this time to the second operand, to include the selected link from the previous page when computing the mean information scent of the current page, changing Eq. 4 to Eq. 5:

$$\begin{aligned} \text{Utility}_{\text{GoBack}} = \quad & \text{MIS(links assessed on previous page} \\ & \quad \text{excluding the selected link)} \\ & - \text{MIS(links assessed on current page} \\ & \quad \textbf{including the selected link from the previous page)} \\ & - \text{GoBackCost} \end{aligned}$$

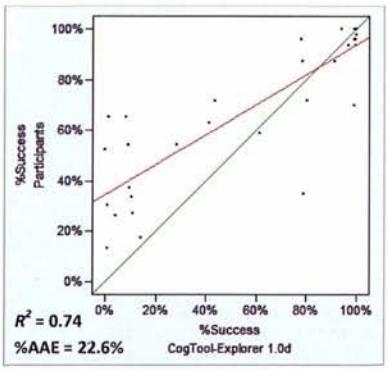
where MIS is Mean Information Scent [Eq. 5]

This second change to the GoBack utility update equation has a nice symmetry with the first change, carrying along the "confidence" inspired by the high infoscent top-level link that was selected. Note that on transiting to a new page, the list of infoscents for the current page (in the second operand of Eq. 5) is still reinitialized to an empty list, but now, the infoscent of the selected link from the previous page will be included when computing the second operand. With Eq. 5, if the infoscent of the selected link from the previous page was very high compared to the other links in the previous page, those other top-level links alone would not exert much pull on the model to go back. If the infoscent of the selected link from the previous page was high relative to the first few links the model sees on the current page, the model would not go back until it "loses confidence" by seeing several low infoscent links in the current page, thereby diluting the effect of the high infoscent link from the previous page that led the model to this current page. CogTool-Explorer 1.0d uses Eq. 5 for its GoBack utility update equation.

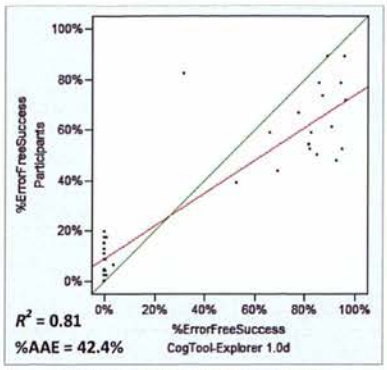
Table 8 presents the results for CogTool-Explorer 1.0d in the multi-page layout (converged after 13 sets of model runs). With the new GoBack utility update equation, CogTool-Explorer 1.0d improved on almost all metrics except for R^2*GoBacks*. Comparing Figures 23a to 22a and 23b to 22b, the new GoBack utility update equation resulted in a movement of those data points above the diagonal (tasks on which participants were successful) towards the right (CogTool-Explorer 1.0a have became as successful).

Table 8: CogTool-Explorer 1.0c and 1.0d compared to participant data in the multi-page layout. The better results are highlighted in bold.

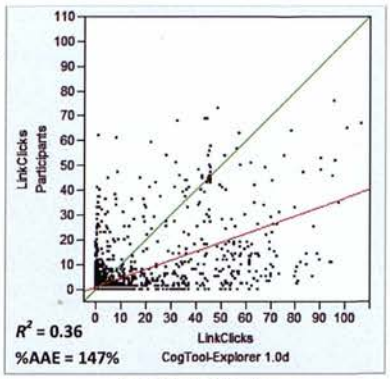
	Correlation, R^2 (95% confidence interval)		%AAE	
	CogTool-Explorer 1.0c	CogTool-Explorer 1.0d	CogTool-Explorer 1.0c	CogTool-Explorer 1.0d
%Success	0.25 (0.18, 0.32)	**0.74** (0.70, 0.78)	40.4%	**22.6%**
%ErrorFreeSuccess	0.59 (0.52, 0.64)	**0.81** (0.78, 0.84)	47.5%	**42.4%**
LinkClicks	0.35 (0.35, 0.36)	**0.36** (0.36, 0.37)	187%	**147%**
GoBacks	**0.31** (0.29, 0.33)	0.27 (0.24, 0.29)	175%	**88.5%**



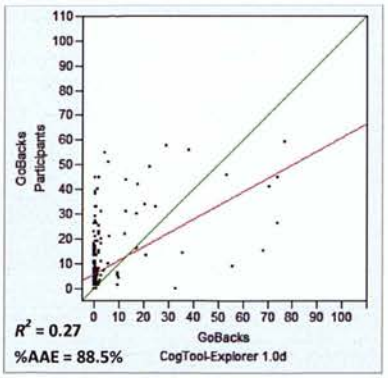
(a) %Success



(b) %ErrorFreeSuccess



(c) LinkClicks



(d) GoBacks

Figure 23: CogTool-Explorer 1.0d compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

Compare the results of CogTool-Explorer 1.0d (Table 8) to the results of CogTool-Explorer 1.0b (Table 4) and it appears that these two models matched participant data in a similar way. In fact, compared to the 26% of go-back actions by participants (Table 6), the percentage of go-back actions by CogTool-Explorer 1.0d dropped to 5%, negating the improvement gained by CogTool-Explorer 1.0c, and is even less than the 6% by CogTool-Explorer 1.0b.

CogTool-Explorer 1.0b has a higher GoBackCost of 5 that makes the model less likely to go back (Eq. 3); CogTool-Explorer 1.0d has a lower GoBackCost of 1 but adds the confidence mechanism in the GoBack utility update equation that also makes the model less likely to go back (Eq. 5). The similar results by both CogTool-Explorer 1.0d and 1.0b may suggest that this is just the replacement of one inhibition factor (higher GoBackCost) by another (confidence mechanism). However, in CogTool-Explorer 1.0d, these changes represent two different influences on the model's go-back behavior: the GoBackCost models the fixed cost incurred from interacting with the UI to go back, while the confidence mechanism makes the model less likely to go back depending on how attractive is the selected link on the previous page that led it to the current page. This confidence is dynamic since the higher the infoscent of the selected link on the previous page, the stronger the inhibition to go back.

While the combined GoBackCost and the confidence mechanism in CogTool-Explorer 1.0d improved the results from CogTool-Explorer 1.0c, it reintroduced the low percentage of go-back actions previously seen in CogTool-Explorer 1.0b. The next section will revisit this problem and introduce a further change to the model that builds on the confidence mechanism.

4.2.1.5.2 Updating of Confidence

Participant data reveal that 26% of participant actions on 2nd-level pages were go-backs, but only 5% of CogTool-Explorer 1.0d actions on 2nd-level pages were go-backs. When participant trials and model runs on the same tasks were inspected in more detail, a pattern common across tasks was found: Participants would visit a 2nd-level page and after selecting a few 2nd-level links that were incorrect, that is, 2nd-level links that did not lead to the correct 3rd-level page, participants would go back to the top-level page. In contrast, for the same task, when CogTool-Explorer 1.0d visited that same 2nd-level page, the model tended to select more incorrect 2nd-level links and persisted longer in that 2nd-level page before going back to the top-level page. It is interesting that Budiu and Pirolli (2007) made a similar observation about go-back behavior by their DOI-ACT model compared to their participant data. DOI-ACT on average took more clicks than participants did to complete their tasks and Budiu and Pirolli explained that this indicates "the backtracking heuristics used by people is not fully captured by the (DOI-ACT) model."

In CogTool-Explorer 1.0d, when the model transits to a 2nd-level page from the top-level page, or when the model goes from a 3rd-level page back to a 2nd-level page, the list of infoscents for the current page (in the second operand of Eq. 5) is reinitialized to an empty list, and the infoscent of the selected top-level link that led the model to this 2nd-level page becomes the confidence factor in the second operand. This meant that each time after the model selected a 2nd-level link that turned out to be incorrect, the model would go back to the 2nd-level page and the GoBack utility reverted to the same value as when the model initially transited to this 2nd-level page after selecting its top-level link.

However, it would be more reasonable to expect that when the selected 2nd-level links turned out to be incorrect, the model should progressively lose confidence about that 2nd-level page, since those selected 2nd-level links had the highest evaluated infoscents, thus, most expected by the model to be the correct link among links that were evaluated in the 2nd-level page. If those high infoscent links in the 2nd-level page turned out to be incorrect, then this 2nd-level page may not be so good after all and the model should progressively be more likely to go back to the top-level page.

The above reasoning led to a further change to the confidence mechanism introduced in CogTool-Explorer 1.0d, to make the model lower its confidence about the current page as the model selects links on that page that turned out to be incorrect. CogTool-Explorer 1.0e implements this change by using Eq. 6 as its GoBack utility update equation. First, consider the inclusion of the selected link from the previous page (in the second operand of Eq. 6) as the confidence factor for the current page. When the model selects a link on the current page that turns out to be incorrect, the model adds one link count with zero infoscent to the confidence factor of the current page, thus lowering the confidence as more links on the current page turned out to be incorrect. A lower confidence will result in a higher GoBack utility, making the model more likely to go back.

$$\begin{aligned} \text{Utility}_{\text{GoBack}} = \; & \text{MIS(links assessed on previous page} \\ & \qquad \text{excluding the selected link)} \\ & - \text{MIS(links assessed on current page} \\ & \qquad \text{including the selected link from the previous page } \mathbf{and} \\ & \qquad \mathbf{incorrect\ links\ from\ the\ current\ page)} \\ & - \text{GoBackCost} \end{aligned}$$

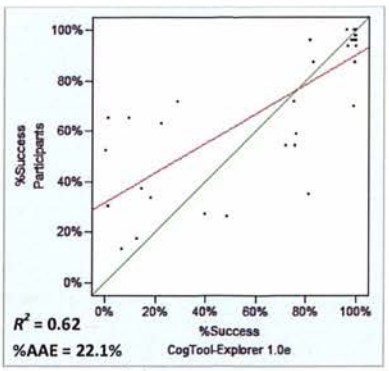
where MIS is Mean Information Scent
and ***incorrect links count as zero information scent*** [Eq. 6]

Note that as explained earlier in this section, the list of infoscents assessed on the current page (in the second operand of Eq. 6) is reinitialized to an empty list each time the model visits or revisits a page. Another detail to note is that the confidence factor for the current page accumulates the counts of incorrect links until the model goes back from the current page, that is, if the model goes from a 2nd-level page back to the top-level page, and later reselects the top-level link and revisits this 2nd-level page, the confidence factor will be initialized to the newly assessed infoscent of the top-level link with no counts of incorrect links.

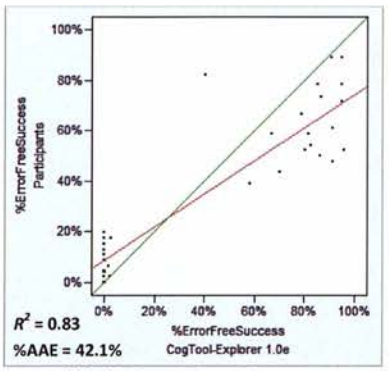
By updating the confidence factor in the GoBack utility update equation, the percentage of go-back actions increased from 5% in CogTool-Explorer 1.0d to 9% in CogTool-Explorer 1.0e, but did not bring it on par with the 26% of go-back actions by participants. Table 9 presents the results for CogTool-Explorer 1.0e in the multi-page layout (converged after 11 sets of model runs). Updating the confidence improved the metrics for GoBacks and all other metrics except R^2*%Success* in CogTool-Explorer 1.0e. Comparing CogTool-Explorer 1.0e to CogTool-Explorer 1.0d on %Success (Figures 24a and 23a), it appears that CogTool-Explorer 1.0e became more successful than participants on some tasks, as seen in the shifting of some data points from the left of the diagonal in Figure 23a to the right of the diagonal in Figure 24a, thus, weakening the linear relationship and lowering R^2*%Success*. The next section will investigate probable causes for why CogTool-Explorer 1.0e became more successful than participants on some tasks.

Table 9: CogTool-Explorer 1.0d and 1.0e compared to participant data in the multi-page layout. The better results are highlighted in bold.

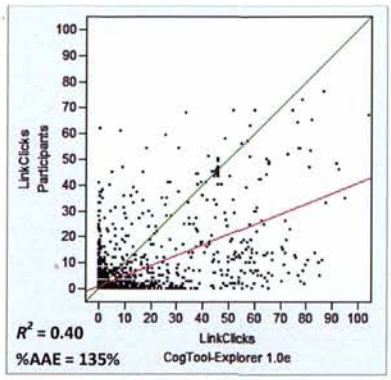
	Correlation, R^2 (95% confidence interval)		%AAE	
	CogTool-Explorer 1.0d	CogTool-Explorer 1.0e	CogTool-Explorer 1.0d	CogTool-Explorer 1.0e
%Success	**0.74** (0.70, 0.78)	0.62 (0.55, 0.67)	22.6%	**22.1%**
%ErrorFreeSuccess	0.81 (0.78, 0.84)	**0.83** (0.80, 0.86)	42.4%	**42.1%**
LinkClicks	0.36 (0.36, 0.37)	**0.40** (0.40, 0.41)	147%	**135%**
GoBacks	0.27 (0.24, 0.29)	**0.41** (0.38, 0.43)	88.5%	**81.3%**



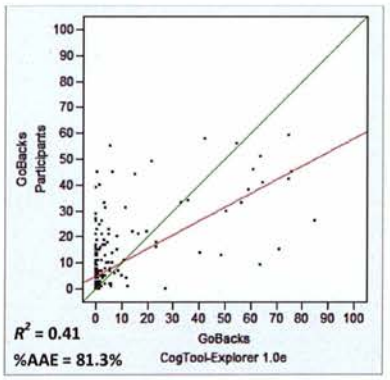
(a) %Success



(b) %ErrorFreeSuccess



(c) LinkClicks



(d) GoBacks

Figure 24: CogTool-Explorer 1.0e compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

4.2.1.6 Alignment to Human-scale Speed of Execution

CogTool-Explorer 1.0e became more successful than participants on a number of tasks, as seen in the shifting of some data points from the left of the diagonal in Figure 23a to the right of the diagonal in Figure 24a. One concern is whether CogTool-Explorer 1.0e was running faster and doing more actions per unit time compared to participants. If so, this would have allowed the model to explore more of the website in the same period of time and increased its task success rate compared to participants. It would also result in the model making more link selections in non-successful model runs as the model continued to make link selections until it ran out of time, exacerbating the overly aggressive selection and reselection of links by the model, as discussed in the later part of Section 4.2.1.4 (after Figure 20).

Participant data reveal that the average duration between link selections by participants was 7.4 seconds, but the average duration between link selections by CogTool-Explorer 1.0e was only 4.8 seconds. Therefore, CogTool-Explorer 1.0e did have a faster speed of execution compared to participants. It is important to rectify this unfair advantage so that the model does not over predict task success rates. The next two sections will describe two flaws that contributed to the problem, one in the mock-up of the multi-page layout, and the other in the CogTool-Explorer 1.0e model, and describe model changes to address these two flaws and align the model speed of execution to better match that of participants.

4.2.1.6.1 Refinement of Perceptual-Motor Requirements in Task

If the modeling of the UI in which the exploration takes place imposed fewer perceptual-motor requirements than it should in reality, it would allow the model to run faster and do more in the same time period. Figure 25 shows the mock-up of the multi-page layout in CogTool-Explorer 1.0e. The mock-up was generated automatically by importing the webpages from the original experiment website. The import function renders and extracts the position, dimension, text and target URL of each link from the actual webpages and creates the mock-up (Section 4.3.1 describes this in more detail). Links in the 3rd-level pages were not included in the mock-up because in a 3rd-level page, links were listed alphabetically and participants could directly check for the target link in a 3rd-level page, thus, unlike in the top-level and 2nd-level pages, assessing the infoscent of links and exploration behavior do not apply in the 3rd-level pages.

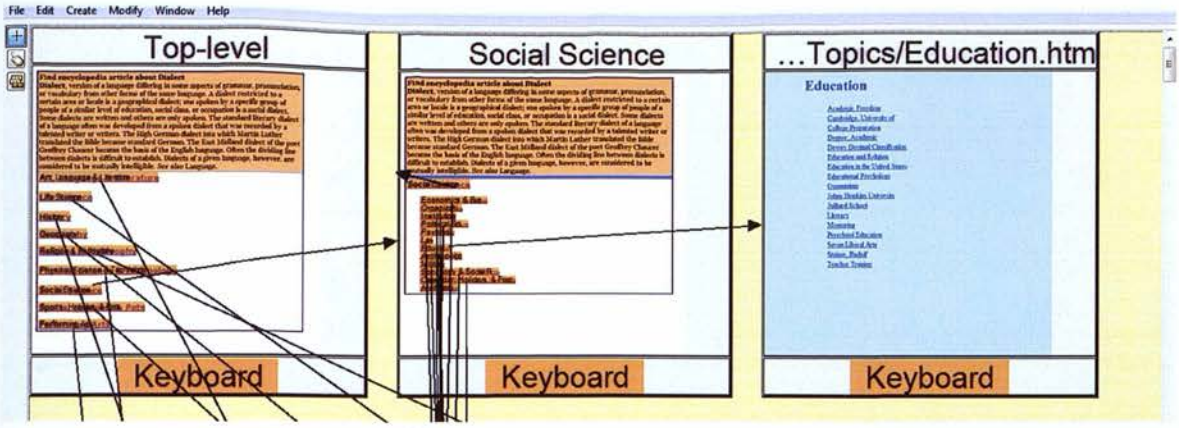


Figure 25: Part of the mock-up of the multi-page layout in CogTool-Explorer 1.0e. The entire mock-up is composed of 103 webpages (one top-level page, nine 2nd-level pages and ninety-three 3rd-level pages). This figure shows the top-level page, a 2nd-level page and a 3rd-level page.

In CogTool-Explorer 1.0e, every time the model transits to a page after selecting a link, software code will compare the name of that new page to the name of the target page specified at the start of the model run, to check if the model has reached the target page that indicates a successful model run. In the multi-page layout, the specified target page will be one of the 3rd-level pages. If the names match, CogTool-Explorer 1.0e will stop. If the names do not match, the model will continue to run and explore that page. If the new page is a 3rd-level page, the model will "see" no links on the page (which does not require any eye movements in the ACT-R vision module) and that will trigger the action to go back to the previous 2nd-level page.

The absence of 3rd-level links in the above mock-up was intended to approximate the deterministic nature of the task when in 3rd-level pages. However, it also removed the perceptual-motor requirement on CogTool-Explorer 1.0e to move its simulated eye to where the target link should be in the page, to check if the target link was actually present or not, just as participants had to do. This simplification of model actions in 3rd-level pages could explain for the shorter duration between link selections by CogTool-Explorer 1.0e compared to participants.

A second perceptual-motor action that participants had to do in the task, but was missing from CogTool-Explorer 1.0e and the mock-up of the multi-page layout, is the need to look at and click on the web browser's go-back button to go back to the previous page. In CogTool-Explorer 1.0e and previous versions of CogTool-Explorer, when the model decides to go back to the previous page, the go-back production will trigger software code to update the model's visual field with the previous page, which is the same software code that updated the model's visual field with a new page when a link is selected. While links are part of the imported webpage and CogTool-Explorer 1.0e will look at a link in the mock-up to click it, the web browser's go-back button is not part of the webpage and thus is not imported into the mock-up. There is no presumption that the task environment is a web browser, for example, the webpages could be a prototype of a mobile phone with its own unique go-back mechanism, thus, go-back buttons are not automatically inserted into the mock-up. However, for our purpose of comparing model behavior to participant data to refine the model, this

simplification of model actions needed for going back to the previous page could explain for the shorter duration between link selections by CogTool-Explorer 1.0e compared to participants.

To rectify the missing perceptual-motor requirements described above, additional UI widgets are added to the mock-up as shown in Figure 26, and additional model productions are added to CogTool-Explorer 1.0f for the procedural knowledge associated with these additional widgets. The first type of widget added is a link widget in each of the 3rd-level pages. When CogTool-Explorer 1.0f transits to a 3rd-level page that is not the target webpage, the existence of at least one link in the page will cause the model to move its simulated eye and look at it, which is the default behavior of the model. These link widgets are specially marked so that after CogTool-Explorer 1.0f looks at the link, new model productions in CogTool-Explorer 1.0f will direct the model to go back to the previous page. While this does not capture the detailed process of looking for the target link at its expected location in the alphabetical list of links on a 3rd-level page, it adds a simple perceptual-motor requirement on CogTool-Explorer 1.0f that approximates what participants have to do in 3rd-level pages.

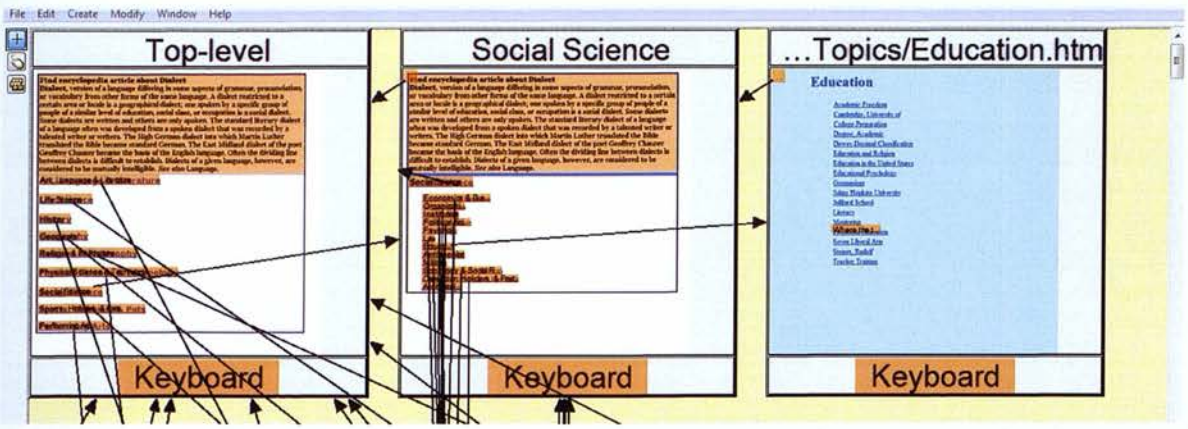


Figure 26: Part of the mock-up of the multi-page layout in CogTool-Explorer 1.0f. The entire mock-up is composed of 103 webpages (one top-level page, nine 2nd-level pages and ninety-three 3rd-level pages). This figure shows the top-level page, a 2nd-level page and a 3rd-level page.

The second type of widget added to the mock-up is a button widget in each of the 2nd-level and 3rd-level pages, positioned at the top left corner of the page as shown in Figure 26. These button widgets are also specially marked so that when the model decides to go back to the previous page, new model productions in CogTool-Explorer 1.0f will direct the model to look at the button widget, move the virtual mouse pointer over the button and click on it. Each of these buttons has a transition that points to its corresponding previous page, and clicking on the button will update the model's visual field with the previous page, which is the default behavior of following a transition. These button widgets simulate the go-back button located at the top left corner of the browser window that participants have to click on to go back to the previous webpage in the experiment.

With the addition of the above UI widgets to the mock-up of the multi-page layout and the additional model productions in CogTool-Explorer 1.0f to enable the model to interact with these added widgets, the average duration between link selections increased from 4.8 seconds in

CogTool-Explorer 1.0e to 5.5 seconds in CogTool-Explorer 1.0f. Although CogTool-Explorer 1.0f now takes longer between link selections, it is still not on par with the 7.4 seconds between link selections by participants. The next section will describe another change in a different part of the model to address this gap.

4.2.1.6.2 Refinement of Duration to Access Information Scent

In CogTool-Explorer 1.0f and earlier versions of CogTool-Explorer, the speed at which the model runs, as indicated by the duration between observable actions like link selections, is determined by the number of model productions that fire between link clicks and the duration that each production took to fire. CogTool-Explorer 1.0f uses the default duration of 50ms for each production firing, which has been the established standard in ACT-R modeling. However, the shorter duration between link selections by CogTool-Explorer 1.0f compared to participants prompts a relook at where in the model the 50ms duration may not be valid.

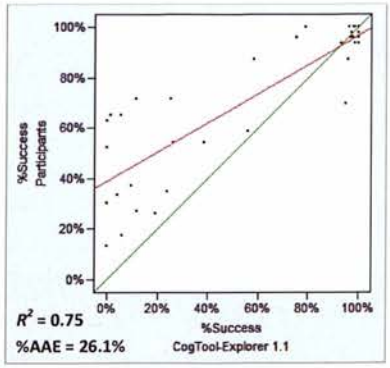
In CogTool-Explorer 1.0f, like in SNIF-ACT 2.0, a sequence of three productions are respectively responsible for looking at a link, encoding the text of the link into the visual buffer, and assessing the infoscent of the encoded link text with respect to the task goal. The first two productions in this sequence, the duration for the production to look at a link, which is determined by EMMA (Salvucci, 2001), and the duration for the encode production, which is the default 50ms, are all well established in the ACT-R modeling literature. The third production to assess the infoscent of a link with respect to the task goal is unique to the CogTool-Explorer and SNIF-ACT 2.0 models. This production will take the encoded text of the link, and the text from the task goal, and invoke a LISP function to compute the infoscent of the link with respect to the goal. The LISP function will then update the utilities of the three competing productions (see Section 4.2.1.1) based on the link's infoscent.

This LISP function is an example of a black-box implementation of the cognitive processes to assess infoscent. A black-box implementation is useful for a computationally efficient implementation of a complex cognitive process, and would sometimes, as is the case for SNIF-ACT 2.0 (Fu & Pirolli, 2007), be accompanied with an explanation of how the black-box implementation could map to an implementation that uses more native ACT-R modeling mechanisms such as spreading activation between declarative memory chunks. However, this substitution meant that in CogTool-Explorer 1.0f, the 50ms duration for the single production that invokes a LISP function to assess infoscent would be shorter than the duration of a more native ACT-R implementation that requires multiple model productions with latencies from declarative memory retrievals.

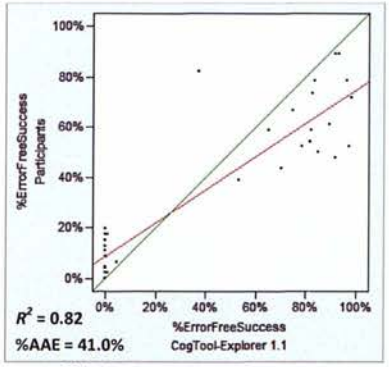
The above reasoning motivated iterative tests of setting longer durations for the assess infoscent production, starting with 100ms steps and narrowing down to 5ms steps. At 275ms for the assess infoscent production in CogTool-Explorer 1.1, the average duration between link selections increased to 7.4 seconds, bringing it on par with the 7.4 seconds by participants. Table 10 presents the results for CogTool-Explorer 1.1 in the multi-page layout (converged after 9 sets of model runs). Comparing CogTool-Explorer 1.1 to CogTool-Explorer 1.0e on *%Success* (Figures 27a and 24a), the longer duration between link selections by CogTool-Explorer 1.1 resulted in lower *%Success* on tasks by the model, as evident from the shifting of data points towards the left in Figure 27a, thus strengthening the linear relationship and increasing R^2*%Success*.

Table 10: CogTool-Explorer 1.0e and 1.1 compared to participant data in the multi-page layout. The better results are highlighted in bold.

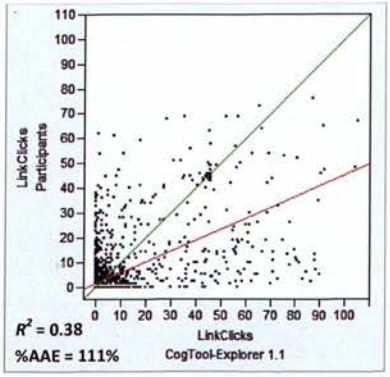
	Correlation, R^2 (95% confidence interval)		%AAE	
	CogTool-Explorer 1.0e	CogTool-Explorer 1.1	CogTool-Explorer 1.0e	CogTool-Explorer 1.1
%Success	0.62 (0.55, 0.67)	**0.75** (0.70, 0.79)	**22.1%**	26.1%
%ErrorFreeSuccess	**0.83** (0.80, 0.86)	0.82 (0.78, 0.85)	42.1%	**41.0%**
LinkClicks	**0.40** (0.40, 0.41)	0.38 (0.37, 0.39)	135%	**111%**
GoBacks	**0.41** (0.38, 0.43)	0.27 (0.24, 0.30)	**81.3%**	84.4%



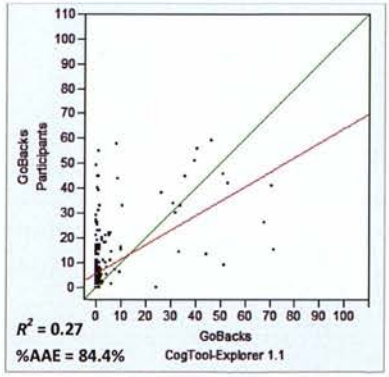
(a) %Success



(b) %ErrorFreeSuccess



(c) LinkClicks



(d) GoBacks

Figure 27: CogTool-Explorer 1.1 compared to participant data in the multi-page layout. Each data point in (a) and (b) represents a task, in (c) a link in a task, and in (d) a 2nd-level page in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

Figure 27b shows an outlier task, the Pigeon task, where *%ErrorFreeSuccess* by participants was 82% but only 37% in CogTool-Explorer 1.1. In fact, the Pigeon task had shown up as an outlier in *%ErrorFreeSuccess* since CogTool-Explorer 1.0a (Figure 19b). In the Pigeon task, the correct top-level link is "Life Sciences" and the correct 2nd-level link is "Birds". The 46 participants selected other top-level links 20% of the time before selecting the correct top-level link "Life Science", and among those other selected top-level links, participants never selected "History". In contrast, CogTool-Explorer 1.1 selected the top-level link "History" 43% of the time before selecting the correct top-level link "Life Science". Why did the model find "History" attractive given the task Pigeon whereas participants never selected "History"?

The infoscent of a link is based on the semantic similarity between the link text and the task goal. The goal for the Pigeon task was:

> "Pigeon, common name for members of a family of birds; smaller species are commonly known as doves, but sizes of pigeons and doves overlap. The birds, almost worldwide in distribution, are most abundant in warm regions. Pigeons have small heads, short necks, stout bodies with short legs, and sleek plumage and have a fleshy or waxy protuberance, the cere, at the base of the bill. They dwell in trees or on the ground and feed on seeds, fruit, acorns and other nuts, and insects. Pigeons fly rapidly and are noted for their cooing call. They build loose, almost flat, nests of twigs, bark, straw, and weeds; the female lays one or two tan or white eggs. The appropriately named white-crowned pigeon, a Caribbean species that extends north to southern Florida, is the only wild member of its genus found in the eastern United States. Among the pigeons called doves, many are widely distributed in Eurasia and Africa."

In CogTool-Explorer 1.0, that is, before the refinement of the infoscent estimate for top-level links, the model selected "Life Science" and "History" only 6% and 8% of the time respectively, because the infoscent of both these links, including the correct link "Life Science", were negative given the task goal! Interestingly, the most selected top-level link was "Geography", which had positive infoscent and was the second highest among the top-level links; the geographic terms in the last two sentences of the task goal contributed to the high infoscent for "Geography". After the refinement of the infoscent estimate for top-level links in CogTool-Explorer 1.0a and later, the infoscents of "History" and "Life Science" are now higher than "Geography", and are in fact the highest and second highest among the top-level links. Why would "History" have a higher infoscent and thus appear more attractive to CogTool-Explorer 1.1 than the correct link "Life Science" given the goal for the Pigeon task?

Further digging reveals that the links in the 2nd-level "History" page is the reason:

- African History
- Ancient History
- European History
- History of Asia and Australasia
- History of the Americas
- People in European History
- People in United States History
- United States History
- World History and Concepts

The text of 2nd-level links "People in United States History" and "United States History" make the top-level link "History" look attractive given the goal for the Pigeon task. In fact, if a small change is made to the task goal by removing the words "United States" near the end of the task goal, the infoscent of "History" will be reduced from being the highest among the top-level links, to less than one-third that of "Life Science", which now has the highest infoscent among the top-level links. The model would then be less likely to select "History" before selecting "Life Science", which would have improved the model's *%ErrorFreeSuccess* in the Pigeon task and moved the outlier in Figure 27b towards to right. This investigation revealed a potential pitfall in computational estimation of infoscent, that even sophisticated and state-of-the-art algorithms like LSA, which have been shown to correlate with human assessments of text semantic similarity and is applied in automated grading of standardized test essays, can occasionally make estimations of infoscent that may be quite different from assessments by people. Since computational estimates of infoscent may not perfectly correlate with human assessments, some number of outliers or mismatches between model behavior and participant data would be expected even for an otherwise perfect model, until further research improve these estimation algorithms.

4.2.1.7 Summary of the Performance of CogTool-Explorer 1.1

Figure 28 plots the results over the seven iterations from CogTool-Explorer 1.0 to CogTool-Explorer 1.1 on the multi-page layout. CogTool-Explorer 1.1 improved on all comparison metrics over CogTool-Explorer 1.0: correlations with participant data have become higher on all four task performance measures and %AAE compared to participant data have become smaller on all four measures.

The improvements in correlations for *%Success* and *%ErrorFreeSuccess* are large: R^2*%Success* more than doubled to 0.75 and R^2*%ErrorFreeSuccess* almost doubled to 0.82. Although there is room for improvement, these values are in the range where UI designers could use them to identify the tasks at the extremes. That is, these metrics identify which tasks are sufficiently supported by the UI that effort can be diverted to other areas and which tasks are in most need of attention. While there are also improvements in %AAE for these two task performance measures, the improvements are modest and not yet within the 10-20% range that Kieras et al. (1997) considered desirable for UI design practice.

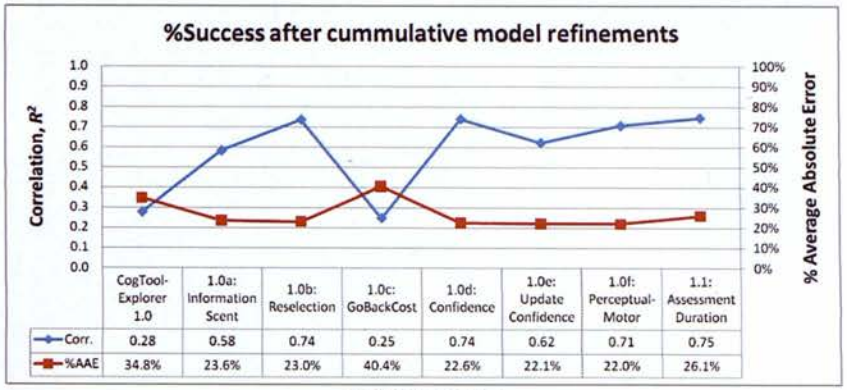
In contrast to the above two task performance measures, the improvement in R^2*LinkClicks* to 0.38 is more modest, and R^2*GoBacks* is practically unchanged at 0.27. While %AAE for LinkClicks is almost halved to 111%, that for GoBacks is practically unchanged at 83.9%, and both are far from the 10-20% range that Kieras et al. (1997) considered desirable for UI design practice. Furthermore, the percentage of go-back actions by CogTool-Explorer 1.1 marginally dropped to 8% (from 9% by CogTool-Explorer 1.0e), which is less than a third of the 26% of go-back actions by participants.

There are two possible explanations for the poorer match by CogTool-Explorer 1.1 to participant data in *LinkClicks* and *GoBacks*. First, as discussed in the later part of Section 4.2.1.4 (after Figure 20), the preponderance of data points to the right of the diagonal in Figure 27c could be accounted for by a free-running model that is less successful than participants. Figure 27a shows that CogTool-Explorer 1.1 was less successful than participants in the majority of the tasks, and in these tasks,

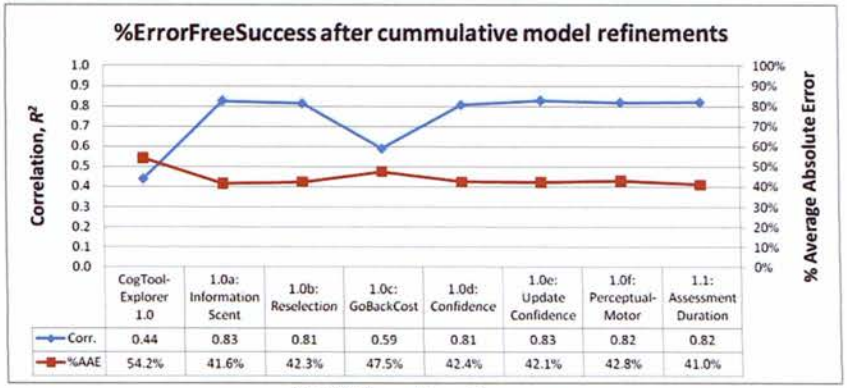
there would be more model runs than participant trials where the free-running CogTool-Explorer 1.1 could continue to select links until the task time was up.

Another possible reason could be due to differences in the estimated infoscent values used by CogTool-Explorer 1.1 versus the perceived infoscent of links by participants. In fact, Blackmon et al. (2002) explicitly noted such "a bias in LSA's similarity estimates with respect to actual user judgment", that "LSA is more likely to overestimate than underestimate the similarity of items", and prescribed "allowing the analyst to reduce LSA's similarity estimate, and reject a proposed competing link label, is an approximate response to this problem." Overestimation of infoscent would make more links look attractive to the model than they actually did to participants and the model would thus select links that participants did not. Overestimation of infoscent would also delay the model from going back to the previous top-level page because more links on 2nd-level pages would look attractive given the task goal. This could account for the lack of go-back actions as seen in the majority of data points to the left of the diagonal in Figure 27d and in the 8% of go-back actions by CogTool-Explorer 1.1 compared to 26% by participants.

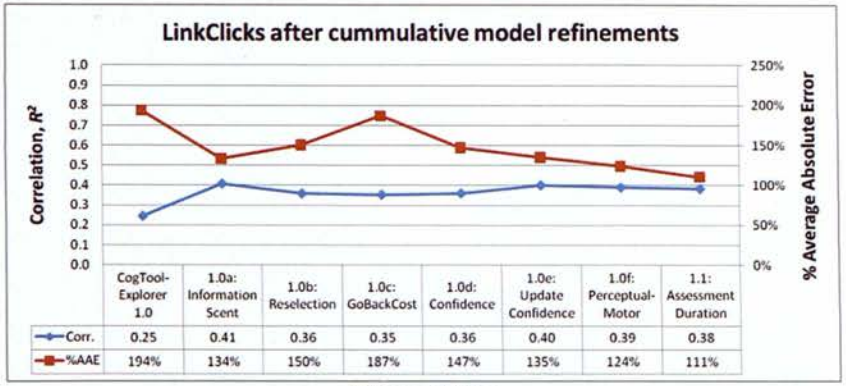
Although there is room for improvement in future research, the relative success of CogTool-Explorer 1.1 compared to CogTool-Explorer 1.0 emboldens the move to the next challenge, the half-flattened layout.



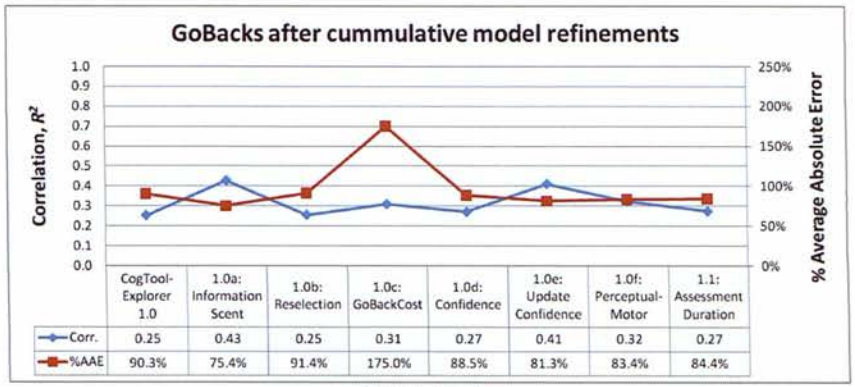
(a) %Success



(b) %ErrorFreeSuccess



(c) LinkClicks



(d) GoBacks

Figure 28: Results over seven iterations from CogTool-Explorer 1.0 to CogTool-Explorer 1.1 in the multi-page layout. The blue diamond markers and the left vertical axis are for Correlation, where higher is better. The red square markers and the right vertical axis are for % Average Absolute Error, where lower is better.

4.2.2 HALF-FLATTEN LAYOUT

The half-flatten layout and tasks in this dissertation are from AutoCWW experiment Expt040423 (Cross-Validation Experiment in Blackmon et al., 2005). In this layout (Figure 29), like in the multi-page layout (Figure 16), the task goal is presented in the paragraph of text under the line "Find encyclopedia article about..." at the top of the top-level and 2nd-level pages. Participants start in the top-level page and on selecting a link, transits to 2nd-level pages. The half-flatten layout differs from the multi-page layout by showing the top-level links in 2nd-level pages, thus, participants can see both top-level links and 2nd-level links in 2nd-level pages. This half-flattened arrangement is akin to the accordion[8] menu pattern in UI design, in that only one top-level link can be expanded at a time. In a 2nd-level page, participants may select a top-level link to go to another 2nd-level page, or may select a 2nd-level link to go to its 3rd-level page (not shown in Figure 29 but shown in Figure 16). In a 3rd-level page, like in the multi-page layout, participants can check that they have succeeded in the task if the target link ("Audiometer" in the example in Figure 29) is in that 3rd-level page, otherwise, participants will go back to the 2nd-level page and continue exploration. Each task has only one correct 3rd-level page that contains the target link, and there is only one correct top-level link and only one correct 2nd-level link that lead to the correct 3rd-level page.

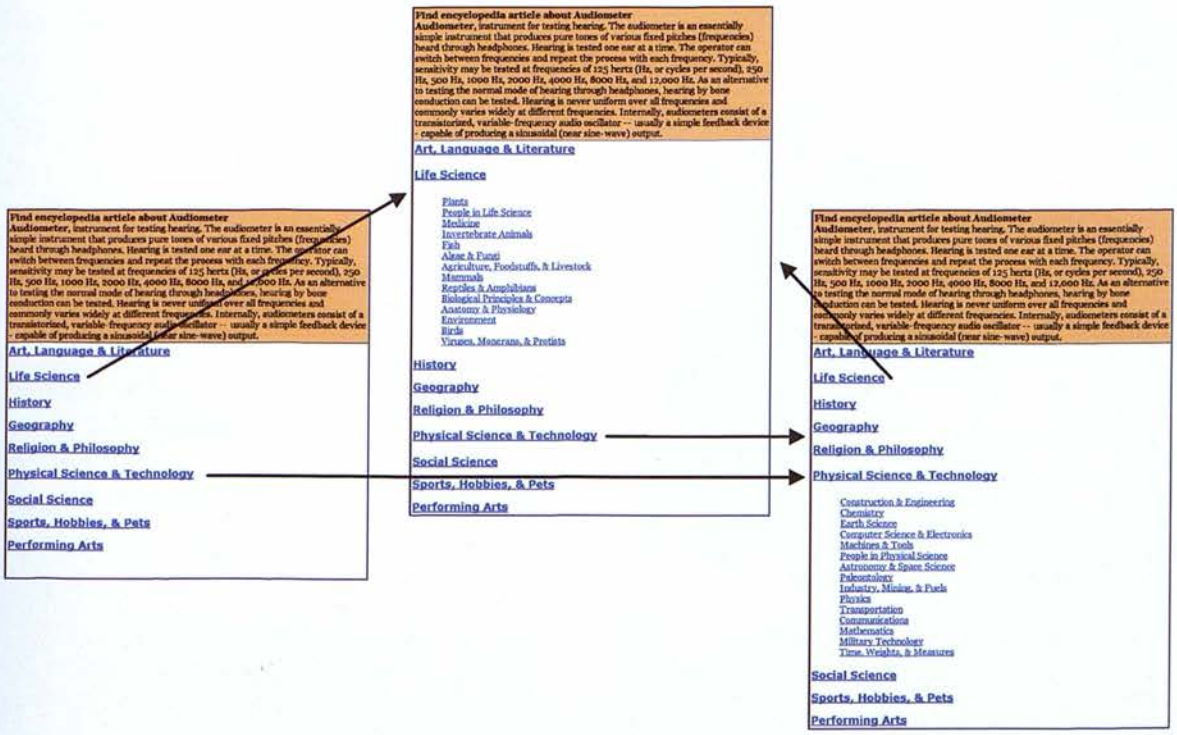


Figure 29: In the half-flatten tasks, participants start in the top-level page (leftmost) and on selecting a link, transits to 2nd-level pages. Participants can see both top-level links and 2nd-level links in 2nd-level pages. Participants may select a top-level link to go to another 2nd-level page, or may select a 2nd-level link to go to its 3rd-level page (not shown here). In a 3rd-level page, participants can check if they have succeeded in the task, and if not, go back to the 2nd-level page and continue exploration.

[8] See for example http://ui-patterns.com/patterns/AccordionMenu

The half-flatten layout differs from the multi-page layout by showing the top-level links in 2nd-level pages. For CogTool-Explorer 1.1, this presents two possible ways for the model to evaluate links in these 2nd-level pages before selecting a link. The first way considers the 2nd-level links that are revealed on transiting to a 2nd-level page to form a group. On transiting to a 2nd-level page, the model first continues exploration among the member links of the group (Figure 30a), and if and when the model decides to go back from the group, it will then continue exploration among the members of the parent group, which are the top-level links and the group of 2nd-level links (Figure 30b). The second way does not differentiate between the 2nd-level links from the top-level links in the same 2nd-level page. On transiting to a 2nd-level page, the model continues exploration among all selectable links, that is, the newly exposed 2nd-level links plus all the top-level links in the 2nd-level page.

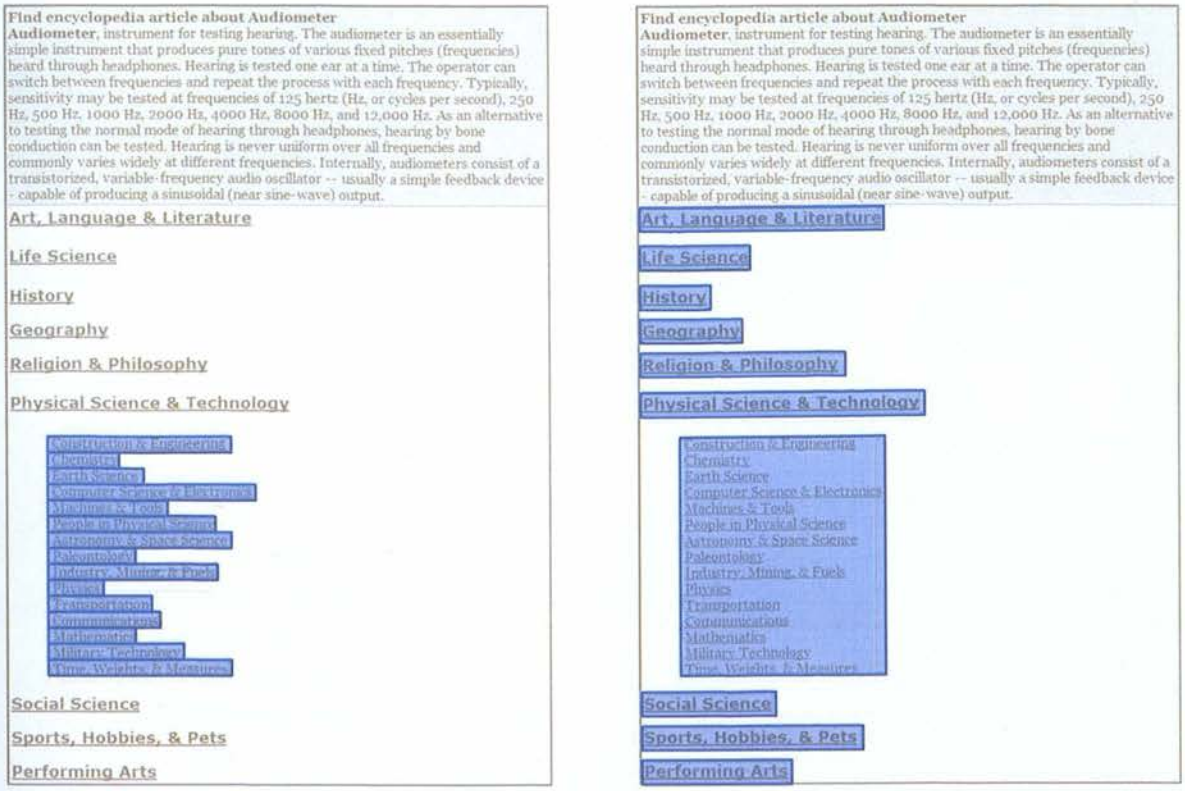


Figure 30: Blue rectangles show the visual elements available at different stages of a group-based hierarchical exploration process in the half-flatten layout. Figure (a) shows on transiting to a 2nd-level page and Figure (b) shows after going back from the group of 2nd-level links.

The thesis of this dissertation hypothesizes that a model that considers groups can make more accurate predictions of user exploration compared to a model that does not, assuming that participants did recognize and utilize the group relationships in the layout. This suggests the first way, which follows the group-based hierarchical exploration process developed from the multi-page layout, when applied to the half-flatten layout, should match participant data better, and

foreshadows a similar result when group-based hierarchical exploration is applied to the multi-group layout with nine groups (Figure 14c). To investigate which of these two ways will result in model behavior that matches participant data better, Section 4.2.2.1 will describe additions made to CogTool-Explorer 1.1 to enable the expression of group relationships in the device model, and for the user model to see and utilize these group relationships during exploration, resulting in CogTool-Explorer 1.2. Sections 4.2.2.2 and 4.2.2.3 will respectively describe in more detail these two ways of operation by CogTool-Explorer 1.2: the first way that considers group relationships (Consider Groups), and the second way that ignores group relationships and treats all links as equals (Ignore Groups). Section 4.2.2.4 will then compare how well Consider Groups and Ignore Groups by CogTool-Explorer 1.2 match participant data.

4.2.2.1 Enable Group Relationships in CogTool-Explorer 1.2

To enable CogTool-Explorer 1.1 to consider group relationships during exploration, two sets of additions are made to CogTool-Explorer 1.1 that resulted in CogTool-Explorer 1.2. The first set of additions are to the device model (region D in Figure 9) to express group relationships in the UI. In the ACT-R vision module, the location of an object in the visual field is represented by a visual-location object (an ACT-R chunk) that has the attributes (ACT-R chunk slots) for on-screen x and y coordinates. The visual-location object also has other slots for basic visual features that are available pre-attentively, such as color and size. As discussed in Section 2.4.2, similarity in basic visual features is one way that groups of visual objects may be perceived and influence visual search, thus, the visual-location object is an appropriate place to express group relationships in the device model.

The additions to the device model comprises of two parts. First, a *member-of-group* slot is added to each visual-location object. Second, a visual-location object is created for each group in the layout. For each visual-location object that belongs to a group, for example a 2nd-level link in the half-flatten layout, the *member-of-group* slot in the 2nd-level link's visual-location object will have a slot value that references the group's visual-location object. Since a group's visual-location object also has a *member-of-group* slot, nested groups can be represented in this scheme. For visual-location objects that do not belong to any group, for example the top-level links and the group of 2nd-level links in the half-flatten layout, their visual-location objects' *member-of-group* slot will have a slot value of *nil*.

The second set of additions are to the user model (region U in Figure 9) to enable CogTool-Explorer 1.2 to see the group relationships now expressed in the device model and follow a group-based hierarchical exploration process. First, a *group-in-focus* slot is added to the ACT-R chunk that represents the model's exploration goal and is initialized to *nil*. Second, model productions were modified to (1) constrain the visual search to visual-location objects with *member-of-group* slots that match the *group-in-focus* slot, (2) when the model decides to select a group to focus on, push the current value of the *group-in-focus* slot into the exploration history and update the *group-in-focus* slot to reference the new group, (3) when the model decides to select a link and transits to a new page, push the current value of the *group-in-focus* slot into the exploration history and update the *group-in-focus* slot to *nil*, and (4) when the model decides to go back, update the *group-in-focus* slot to the most recent entry in the exploration history, and then delete that most recent entry in

the exploration history. Existing code in CogTool-Explorer 1.2 correctly handles the two cases that occur in the half-flatten layout, where the decision to go back will lead to a previous group in the same page and where the decision to go back will lead to a previous page. Sections 4.2.2.2 and 4.2.2.3 illustrates how these two sets of additions operate in the half-flatten layout.

As explained in Section 3.2, this dissertation does not attempt to investigate the psychological processes of how visual groups are formed and recognized, but rather focuses on how groups affect exploration. Thus, group relationships are provided as input to CogTool-Explorer 1.2 from either the human modeler, as is the case in this dissertation (Section 4.3.2 describes this in more detail) and other prior research (for example, AutoCWW by Blackmon et al., 2002, 2003, 2005; Halverson & Hornof, 2006, 2007, 2008; and DOI-ACT by Budiu & Pirolli, 2007), or in future work from other computational models of visual grouping (for example, Rosenholtz, Twarog, Schinkel-Bielefeld & Wattenberg, 2009).

4.2.2.2 Consider Groups: Operation of CogTool-Explorer 1.2

Figure 31 illustrates an example run of Consider Groups by CogTool-Explorer 1.2 in the half-flatten layout. Note that steps 1 to 5a in Figure 31 follows CogTool-Explorer 1.0 in the two-column layout as presented in Figure 10 and Sections 4.1.1 to 4.1.4, and CogTool-Explorer 1.1 in the multi-page layout as presented in Figure 17 and Section 4.2.1.1, thus this section will start from step 5b where the model transits to a 2nd-level page in the half-flatten layout.

On transiting to a 2nd-level page, the model's *group-in-focus* slot is set to reference the group of 2nd-level links (step 5b). The model will request the ACT-R vision module for unattended visual-location objects with matching *member-of-group* slots, thus, exploration is constrained to the member links of the group (Figure 30a). The model will look at the nearest link in the group (Figure 31, step 5c), evaluate the link's infoscent (step 5d) and remember the link as the best widget (can be a link or a group, but only links apply in this stage of this example as there are no nested groups) if it has the highest infoscent so far in the page, and then decide to either look at and evaluate another link (step 5e), or stop and select the best link seen so far in the group (step 6), or go back from the group (step 7). This three-way decision uses the same utility update equations and parameter values as CogTool-Explorer 1.1.

If and when CogTool-Explorer 1.2 decides to go back from the group of 2nd-level links, the *group-in-focus* slot will be set to *nil*, and its memory of the best widget so far (can be a link or a group, and both apply in this stage of this example) will be reset to be updated with a new best going forward. The model will then continue exploration, now constrained to the top-level links and the group of 2nd-level links (Figure 30b), making the same three-way decision after each evaluation. In this example, the model decided to stop and select the best link after evaluating several links (Figure 31, step 8).

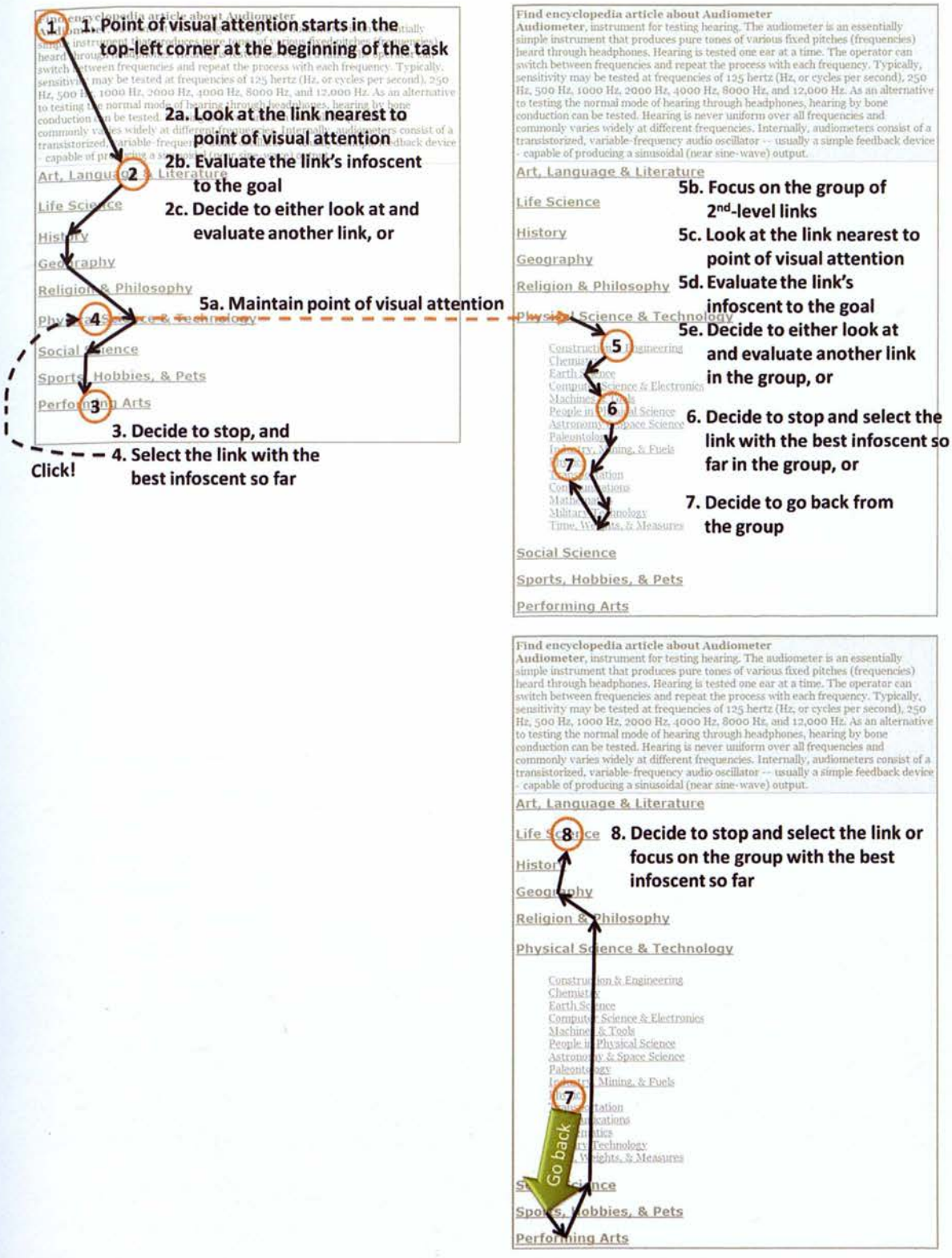


Figure 31: An example run of Consider Groups by CogTool-Explorer 1.2 in the half-flatten layout.

There are four additional details about the operation of CogTool-Explorer 1.2 in the half-flatten layout. First, the default behavior of CogTool-Explorer 1.2 is to start exploration on a new page with the *group-in-focus* slot set to *nil*, which is appropriate when the new page is different from the previous page. However, in the half-flatten layout, the top-level and 2nd-level pages are actually continuations of one another: transiting to a 2nd-level page reveals the 2nd-level links associated with the selected top-level link, and the appropriate model behavior is to continue exploration within the group of 2nd-level links. Thus, on visiting a 2nd-level page, the model knows to focus on the group of 2nd-level links in that page and continue its exploration there. This specific knowledge is implemented by an implicit focus on the group upon transiting to a 2nd-level page, by pushing the initial *nil* value of the *group-in-focus* slot into the exploration history and updating the *group-in-focus* slot to reference the group of 2nd-level links.

Second, on a basis of minimal assumption, the default reselection behavior of CogTool-Explorer 1.2 is to treat all links in the layout as unique. However, in the half-flatten layout, the top-level and 2nd-level pages are actually continuations of one another: the top-level links that have the same display labels in the top-level and 2nd-level pages are actually the same link. Therefore, the model should treat them as the same link with respect to the reselection behavior (Section 4.2.1.4). Thus, when the model attempts to retrieve from memory if a top-level link had been previously selected, for example "Physical Science and Technology" in the 2nd-level page (Figure 31), an ACT-R chunk that represents the "Physical Science and Technology" link in the top-level page that had its *status* slot set to *chosen* will match and may be retrieved. This specific knowledge is implemented by matching on the common substring from the *widget-name* slot (in this example, the substring "Physical Science and Technology") when the model attempts to retrieve from memory if a top-level link had been previously selected, instead of matching on the entire string from the *widget-name* slot that is unique for every link in the device model.

Third, when CogTool-Explorer 1.2 chooses to focus on a group of 2nd-level links, or chooses to select the top-level link associated with the group, the resulting model behaviors are the same: exploration continues among the member links of the group. Therefore, the model should treat these two actions as the same. Thus, when the model attempts to retrieve from memory whether the "Physical Science and Technology" group had been previously focused on, an ACT-R chunk that represents the "Physical Science and Technology" link that had its *status* slot set to *chosen* will match and may be retrieved. Like in the previous paragraph, this specific knowledge is also implemented by matching on the common substring in the *widget-name* slot when the model attempts to retrieve from memory if a group had been previously focused on, instead of matching on the entire *widget-name* slot that is unique for every link and group in the device model. Furthermore, in the half-flatten layout, the text that is assembled (Section 4.3.2.1 describes this in detail) to compute the infoscent of a group is the same assembled text used to compute the infoscent of its top-level link as described in Section 4.2.1.3.

Finally, on a 2nd-level page in this layout, the top-level link associated with the group is modeled as a Remote Label of the group (Section 4.3.2.1 describes Remote Labels in detail) because it is a heading that describes its associated group. Although the visual-location object representing the heading link has its *member-of-group* slot set to *nil*, which matches the model's *group-in-focus* slot,

the model will not consider it as a potential link to look at after going back from the group in a 2nd-level page. This specific knowledge is implemented by making Remote Labels not match the model's request to the ACT-R vision module for unattended visual-location objects. While this way of modeling a group and its heading link is a simplification of how people would look at and relate the heading link to its group, it meant that CogTool-Explorer 1.2 would see nine options (one group and the other eight top-level links) after going back from the group in a 2nd-level page; the same number of options (nine top-level links) as in the top-level page of the multi-page layout. Future work, with the availability of human eye-tracking data, can investigate and model how people relate the heading link to its group in more detail.

4.2.2.3 Ignore Groups: Operation of CogTool-Explorer 1.2

Figure 32 illustrates an example run of Ignore Groups by CogTool-Explorer 1.2 in the half-flatten layout. Note that steps 1 to 5a in Figure 32 are identical to those in Figure 31 and were previously explained, thus, this section will start from step 5b where the model transits to a 2nd-level page in the half-flatten layout.

It is very easy to make CogTool-Explorer 1.2 ignore groups in the half-flatten layout. First, the group relationships in the 2nd-level pages are simply removed or not modeled in the first place (Section 4.3.2 describes this in detail), thus, all the links' visual-location object have their *member-of-group* slot set to *nil*. Second, the implicit focus on the group upon transiting to a 2nd-level page that was added in Consider Groups is removed, reverting CogTool-Explorer 1.2 to its default behavior of starting exploration on a new page with the *group-in-focus* slot set to *nil*. Without any links belonging to a group and no groups to focus on, CogTool-Explorer 1.2 will simply explore the half-flatten layout in the same manner as in the multi-page layout (compare Figure 31 to Figure 17). On transiting to a 2nd-level page, the model will proceed as it did on the previous top-level page: it will look at the nearest link (step 5b), evaluate the link's infoscent (step 5c) and remember the link as the best link if it has the highest infoscent so far in the page, and then decide to either look at and evaluate another link (step 5d), or stop and select the best link seen so far on the page (step 6), or go back to the previous page (step 7). Like in Consider Groups, this three-way decision uses the same utility update equations and parameter values as CogTool-Explorer 1.1.

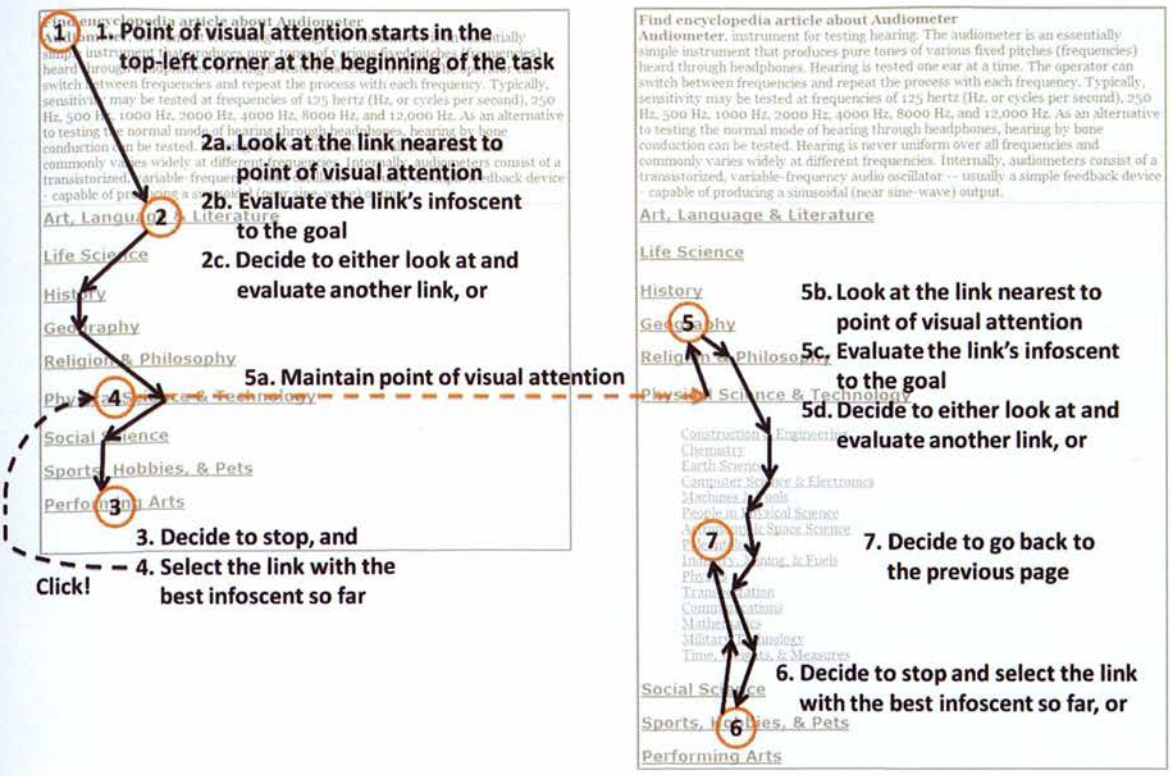


Figure 32: An example run of Ignore Groups by CogTool-Explorer 1.2 in the half-flatten layout.

4.2.2.4 Test of CogTool-Explorer 1.2

To test CogTool-Explorer 1.2, the comparison metrics were computed from model runs in the half-flatten layout and from participant data in the AutoCWW experiment. That experiment included the same 36 tasks from the multi-page layout, that is, the same task goal and correct 3rd-level page, except now the tasks are performed in the half-flatten layout. Participants had 130 seconds to complete each task, failing which the trial is considered a failure. There were 51 to 60 valid participant trials recorded for each task.

Table 11 presents the results for both Consider Groups (converged after 15 sets of model runs) and Ignore Groups (converged after 10 sets of model runs) by CogTool-Explorer 1.2 in the half-flatten layout and Figure 33 shows the scatter plots where each data point represents a task (Figures 33a to 33d) or a link in a task (Figures 33e and 33f). Results for the measure *GoBacks* are not in this analysis because as explained in section 4.2.1, go-back actions by participants are eye movements and were not recorded in the participant log files. Results that show Consider Groups matching participant data better than Ignore Groups would support the thesis of this dissertation. However, the results are mixed. For correlation, Consider Groups is only marginally better than Ignore Groups as indicated by R^2*%Success* (**0.68** versus **0.66**) and R^2*%ErrorFreeSuccess* (**0.82** versus **0.81**), but Consider Groups is worse than Ignore Groups as indicated by R^2*LinkClicks* (**0.38** versus **0.48**). For %AAE, Consider Groups is worse than Ignore Groups on all three measures: *%Success* (**27.0%** versus **26.2%**), *%ErrorFreeSuccess* (**41.3%** versus **33.8%**) and *LinkClicks* (**135%** versus

105%). These mixed results do not give clear support for the thesis of this dissertation. On a positive note, the same model parameters used in CogTool-Explorer 1.1 for the multi-page layout transferred to CogTool-Explorer 1.2 for the half-flatten layout without worsening the fit between model and participants very much; the results from CogTool-Explorer 1.2 (Table 11) are comparable to the results from CogTool-Explorer 1.1 (Table 10). This suggests that the model parameters are not over fitted to a particular layout and do work in another layout.

For CogTool-Explorer 1.2, the mixed results in the half-flatten layout suggest that a group-based hierarchical exploration process did not provide a clear advantage. One possible explanation may be that in the half-flatten layout, the number of links and groups in the 2nd-level page is not large enough for a group-based process to make a clear difference. Section 4.2.3 will investigate this by running CogTool-Explorer 1.2 on a layout with many more visible links.

Table 11: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the half-flatten layout. The better results are highlighted in bold.

	Correlation, R^2 (95% confidence interval)		%AAE	
	Consider Groups	**Ignore** Groups	**Consider** Groups	**Ignore** Groups
%Success	**0.68** (0.64, 0.72)	0.66 (0.60, 0.71)	27.4%	**26.4%**
%ErrorFreeSuccess	**0.82** (0.79, 0.84)	0.81 (0.78, 0.84)	41.7%	**33.6%**
LinkClicks	0.38 (0.37, 0.38)	**0.48** (0.47, 0.49)	135%	**105%**

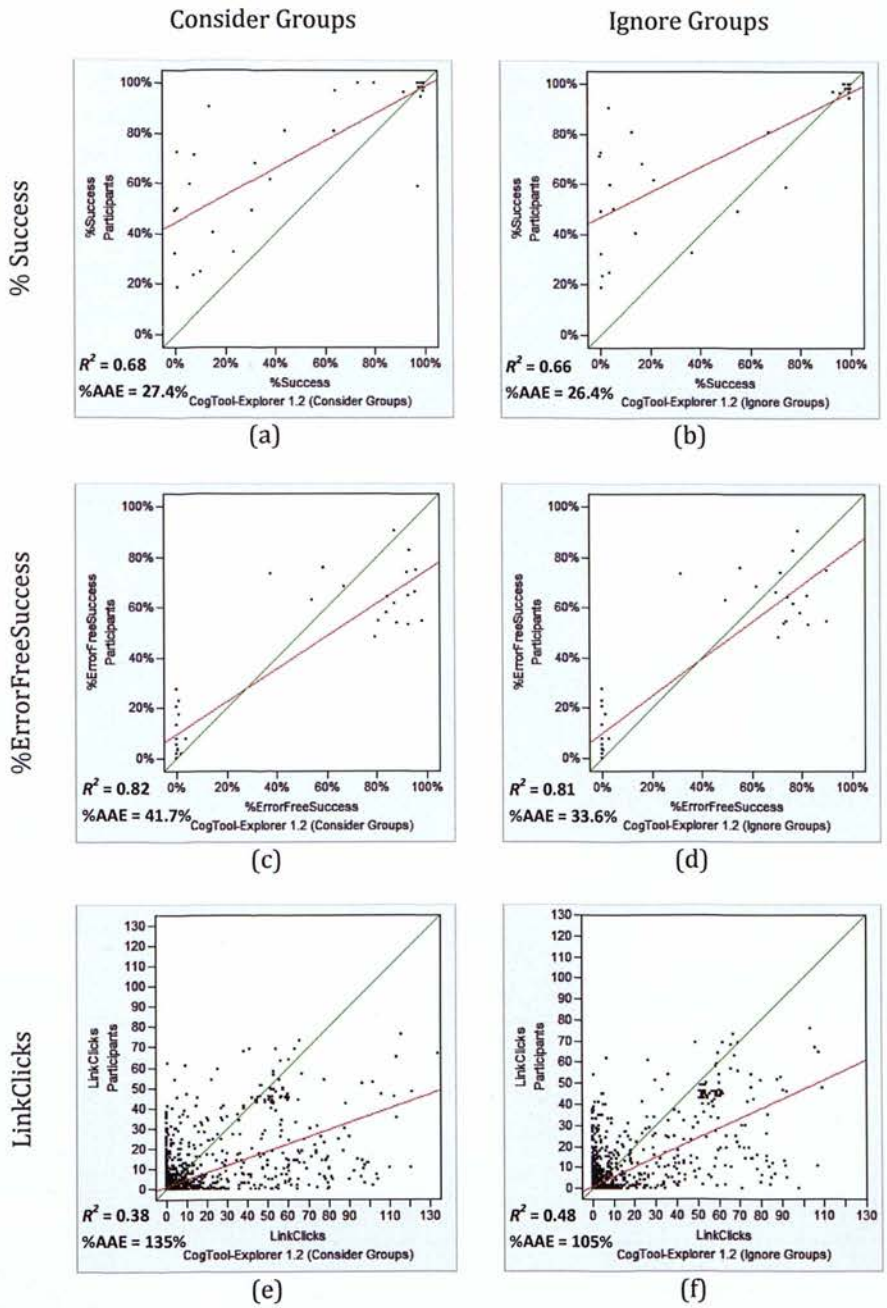


Figure 33: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the half-flatten layout. Each data point in (a) (b) (c) and (d) represents a task, and in (e) and (f) a link in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

4.2.3 MULTI-GROUP LAYOUT

The multi-group layout and tasks in this dissertation are from AutoCWW experiment Expt100419. In this layout (Figure 34), like in the multi-page and half-flatten layouts (Figures 16 and 29), the task goal is presented in the paragraph of text under the line "Find encyclopedia article about..." at the top of the page. Unlike the previous two layouts, the multi-group layout completely flattens the top-level and 2nd-level pages, so all 2nd-level links are visible on the top-level page and there are no 2nd-level pages. Selecting a link in the multi-group layout transits to 3rd-level pages like those in the previous two layouts (not shown in Figure 34 but shown in Figure 16). In a 3rd-level page, like in the previous layouts, participants can check that they had succeeded in the task if the target link ("Audiometer" in the example in Figure 34) is in that 3rd-level page, otherwise, participants will go back to the top-level page and continue exploration. Each task has only one correct 3rd-level page that contains the target link, and there is only one correct link in the top-level page that will lead to that correct 3rd-level page.

Like the half-flatten layout in section 4.2.2, the multi-group layout presents two possible ways for CogTool-Explorer 1.2 to evaluate links in the top-level page before selecting a link. In Consider

Find encyclopedia article about Audiometer

Audiometer, instrument for testing hearing. The audiometer is an essentially simple instrument that produces pure tones of various fixed pitches (frequencies) heard through headphones. Hearing is tested one ear at a time. The operator can switch between frequencies and repeat the process with each frequency. Typically, sensitivity may be tested at frequencies of 125 hertz (Hz, or cycles per second), 250 Hz, 500 Hz, 1000 Hz, 2000 Hz, 4000 Hz, 8000 Hz, and 12,000 Hz. As an alternative to testing the normal mode of hearing through headphones, hearing by bone conduction can be tested. Hearing is never uniform over all frequencies and commonly varies widely at different frequencies. Internally, audiometers consist of a transistorized, variable-frequency audio oscillator—usually a simple feedback device—capable of producing a sinusoidal (near sine-wave) output.

Sports, Hobbies, & Pets	Performing Arts	Religion & Philosophy
Sports Sports Figures Games, Hobbies, & Recreation Pets	Theater Musicians & Composers Cinema, Television, & Broadcasting Music Dance Musical Instruments	Theology & Practices Mythology Religious Figures Philosophy Religions & Religious Groups Scripture The Occult
Art, Language & Literature	**Geography**	**History**
National & Regional Literature Literature & Writing Architecture Artists Language Writers & Poets Decorative Arts Legends & Folklore National & Regional Art Painting, Drawing, & Graphic Arts Sculpture Periods & Styles Photography	World Cities, Towns, & Villages Regions of the World Rivers, Lakes, & Waterways Parks & Monuments Countries Canadian Provinces & Cities Islands Mountain Ranges, Peaks, & Landforms U.S. Cities, Towns, & Villages Maps & Mapmaking Oceans & Seas Exploration & Explorers U.S. States, Territories, & Regions	History of Asia & Australasia People in European History People in United States History United States History African History World History & Concepts Ancient History History of the Americas European History
Physical Science & Technology	**Life Science**	**Social Science**
Construction & Engineering Chemistry Earth Science Computer Science & Electronics Machines & Tools People in Physical Science Astronomy & Space Science Paleontology Industry, Mining, & Fuels Physics Transportation Communications Mathematics Military Technology Time, Weights, & Measures	Plants People in Life Science Medicine Invertebrate Animals Fish Algae & Fungi Agriculture, Foodstuffs, & Livestock Mammals Reptiles & Amphibians Biological Principles & Concepts Anatomy & Physiology Environment Birds Viruses, Monerans, & Protists	Economics & Business Organizations Institutions Political Science Psychology Law Education Anthropology Military Sociology & Social Reform Calendar, Holidays, & Festivals Archaeology

Figure 34: In the multi-group tasks, participants can see all nine groups of links on the top-level page. Selecting a link will transit to 3rd-level pages like those in the multi-page and half-flatten layouts (not shown here). In a 3rd-level page, participants could check if they had succeeded in the task, and if not, go back to the top-level webpage and continue exploration.

Groups, the model will start by evaluating groups and select a group to focus on, and then continue exploration among the member links of that group. If and when the model decides to go back from that group, it will revert to evaluating groups and select another group to focus on. In Ignore Groups, the model will explore among all selectable links in the page, without following the group-based hierarchical exploration process.

The thesis of this dissertation hypothesizes that a model that considers groups can make more accurate predictions of user exploration compared to a model that does not, assuming that participants did recognize and utilize the group relationships in the layout. However, in the previous test on the half-flatten layout in section 4.2.2.4, results were mixed between Consider Groups and Ignore Groups. One possible explanation is that the number of links and groups in the half-flatten layout is relatively small, compared to the number of links and groups in the multi-group layout. To investigate this, Sections 4.2.3.1 and 4.2.3.2 will respectively describe the operation of Consider Groups and Ignore Groups by CogTool-Explorer 1.2 in the multi-group layout. Section 4.2.3.3 will then compare how well Consider Groups and Ignore Groups by CogTool-Explorer 1.2 match participant data.

4.2.3.1 Consider Groups: Operation of CogTool-Explorer 1.2

Figure 35 illustrates an example run of Consider Groups by CogTool-Explorer 1.2 in the multi-group layout. As explained in Section 4.2.2.1, each of the nine groups in the layout is represented by a visual-location object with its *member-of-group* slot set to *nil*, and each of the 93 links has its *member-of-group* slot reference its group's visual-location object. The model's point of visual attention starts in the top-left corner of the page (step 1) and will request the ACT-R vision module for unattended visual-location objects with *member-of-group* slots matching the model's *group-in-focus* slot. With the *group-in-focus* slot initialized to *nil*, the model will begin exploration among the nine groups. The model will look at the nearest group (step 2a), evaluate the group's infoscent (step 2b) and remember the group as the best widget if it has the highest infoscent so far in the page, and then decide to either look at and evaluate another group (step 2c), or stop and focus on the best group seen so far (step 3). To compute the infoscent scores for the nine groups, the text that is assembled (Section 4.3.2.1 describes this in detail) for each of the nine groups is the same as the assembled text for each of the corresponding nine groups in the half-flatten layout, thus, the corresponding groups in the multi-group layout and in the half-flatten layout have the same computed infoscent score.

When the model decides to focus on the best group, it will update its *group-in-focus* slot to reference that group (Step 4) and its memory of the best widget so far will be reset to be updated with a new best going forward. CogTool-Explorer 1.2 will then request the ACT-R vision module for unattended visual-location objects with *member-of-group* slots matching the model's updated *group-in-focus* slot, thus, exploration is now constrained to the member links of the focused group. The model will look at the nearest link in the group (step 5a), evaluate the link's infoscent (step 5b) and remember the link as the best widget if it has the highest infoscent so far in the group, and then decide to either look at and evaluate another link (step 5c), or stop and select the best link seen so far in the group (step 6), or go back from the group (step 7). This three-way decision uses the same utility update equations and parameter values as CogTool-Explorer 1.1.

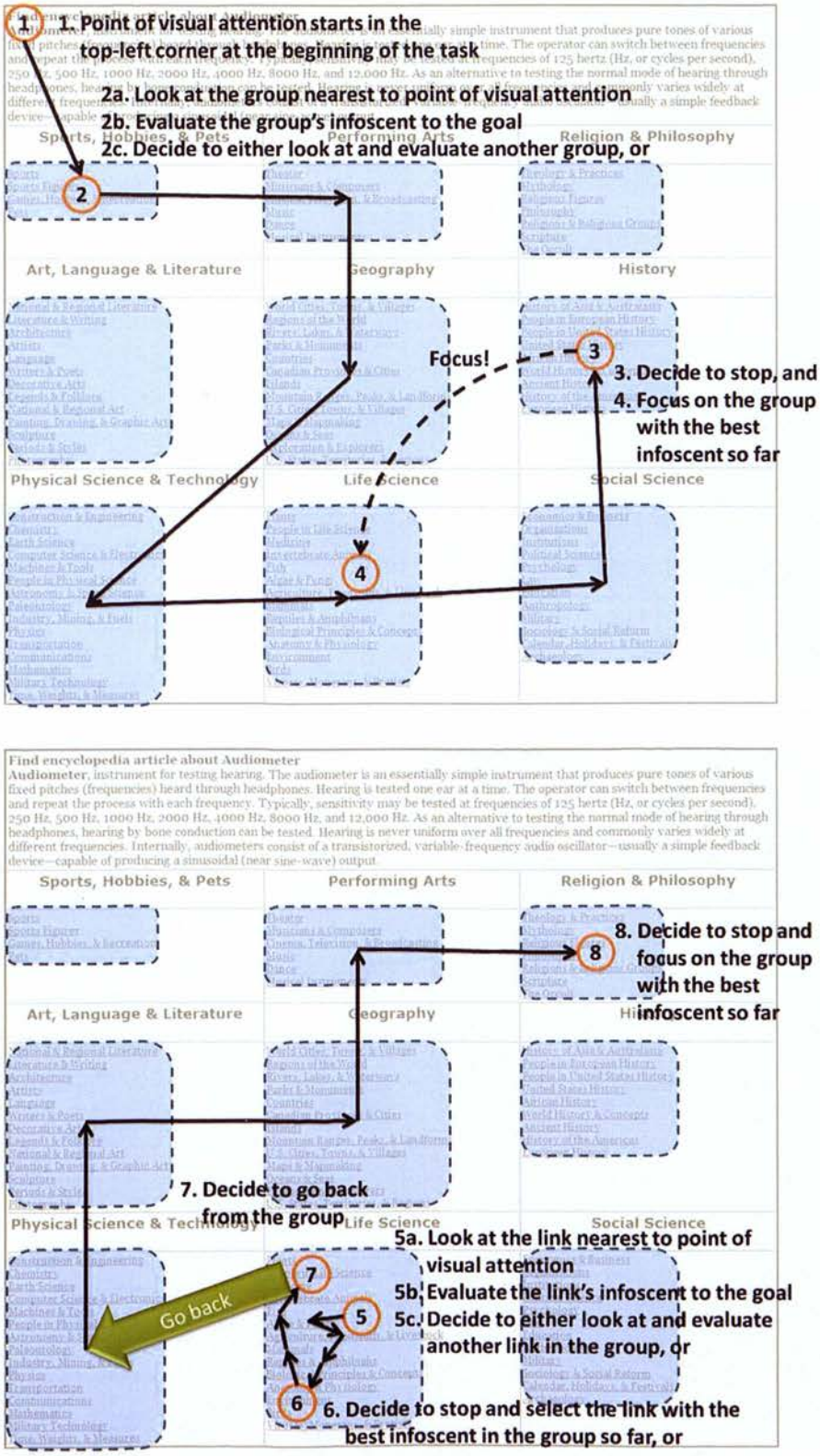


Figure 35: An example run of Consider Groups by CogTool-Explorer 1.2 in the multi-group layout.

If and when CogTool-Explorer 1.2 decides to go back from the group, the *group-in-focus* slot will be set to the most recent entry in the exploration history, which in this case is *nil*. The model will then continue exploration, now back amongst the nine groups. In this example, the model decided to stop and focus on the best group after evaluating several groups (step 8).

There is one additional detail about the operation of CogTool-Explorer 1.2 in the multi-group layout. In the actual experiment website, the nine top-level headings (Sports Hobbies, & Pets, Performing Arts, etc) are not clickable links, but are non-interactive text labels. In the UI mockup, these nine headings are modeled as Text Label widgets, and are Remote Labels of their respective groups (Section 4.3.2.1 describes Remote Labels in detail) because each is a heading that describes its associated group. Although the visual-location objects representing these nine widgets have their *member-of-group* slots set to *nil*, which matches the model's *group-in-focus* slot, the model will not consider them as potential widgets to be looked at separately from their associated groups. This specific knowledge is implemented by making Remote Labels not match the model's request to the ACT-R vision module for unattended visual-location objects. While this way of modeling a group and its text heading is a simplification of how people would look at and relate the text heading to its group, it meant that CogTool-Explorer 1.2 would see nine groups of links in the multi-group layout; the same number of options as in the top-level of both the multi-page and half-flatten layouts. Future work, with the availability of human eye-tracking data, can investigate and model how people relate the heading to its group in more detail.

4.2.3.2 Ignore Groups: Operation of CogTool-Explorer 1.2

Figure 36 illustrates an example run of Ignore Groups by CogTool-Explorer 1.2 in the multi-group layout. Like Ignore Groups on the half-flatten layout in section 4.2.2.3, it is very easy to make the model ignore groups in the multi-group layout: the group relationships are simply removed or not modeled in the first place (Section 4.3.2 describes this in detail), thus, all the links' visual-location objects have their *member-of-group* slot set to *nil*. Without any links belonging to a group and no groups to focus on, the model will simply explore the multi-group layout in the same manner as in the multi-page layout (compare Figure 36 to Figure 17) and the Ignore Groups case in the half-flatten layout (compare Figure 36 to Figure 32). Steps 1 to 4 in Figure 36 follow the same process as previously described for those two other layouts.

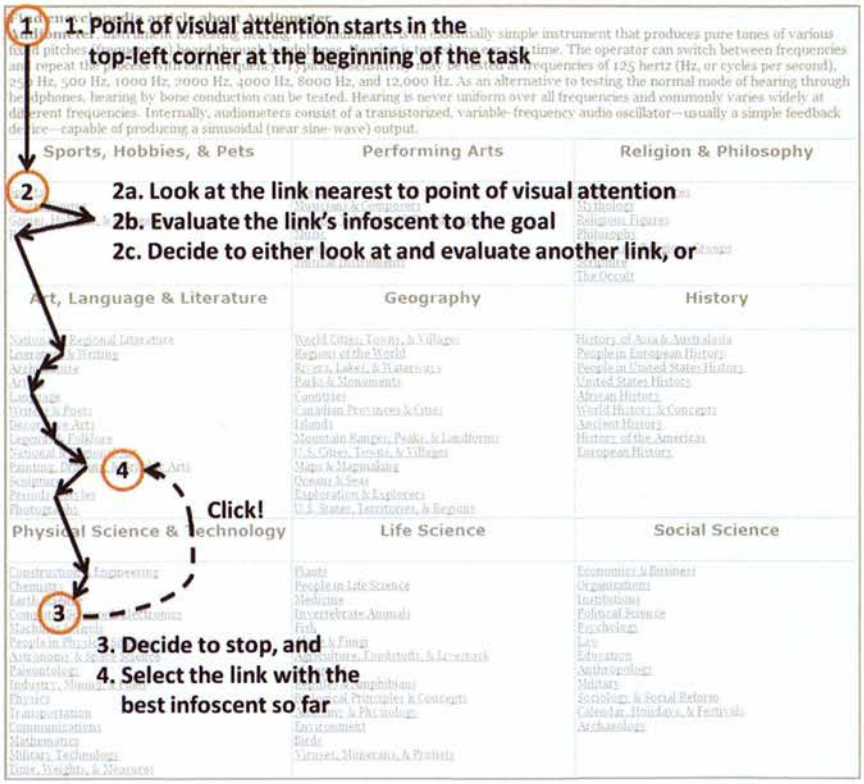


Figure 36: An example run of Ignore Groups by CogTool-Explorer 1.2 in the multi-group layout.

4.2.3.3 Test of CogTool-Explorer 1.2

To test CogTool-Explorer 1.2, the comparison metrics were computed from model runs in the multi-group layout and from participant data in the AutoCWW experiment. That experiment included the same 36 tasks from the multi-page and half-flatten layouts, that is, the same task goal and correct 3rd-level page, except now the tasks are performed in the multi-page layout. Participants had 130 seconds to complete each task, failing which the trial is considered a failure. There were 36 valid participant trials recorded for each task.

Table 12 presents the results for both Consider Groups (converged after 16 sets of model runs) and Ignore Groups (converged after 15 sets of model runs) by CogTool-Explorer 1.2 in the multi-group layout and Figure 37 shows the scatter plots where each data point represents a task (Figures 37a to 37d) or a link in a task (Figures 37e and 37f). As explained in section 4.2.2.4, results for the measure *GoBacks* are not in this analysis because go-back actions by participants are eye movements and were not recorded in the participant log files. For correlation, Consider Groups is better than Ignore Groups on all three measures: *%Success* (**0.67** versus **0.46**) and *%ErrorFreeSuccess* (**0.72** versus **0.34**), and *LinkClicks* (**0.33** versus **0.25**). For %AAE, Consider Groups is also better than Ignore Groups on all three measures: *%Success* (**30.2%** versus **33.4%**), *%ErrorFreeSuccess* (**45.6%** versus **84.2%**) and *LinkClicks* (**115%** versus **133%**).

In contrast to the mixed results between Consider Groups and Ignore Groups in the half-flatten layout, here in the multi-group layout, Consider Groups matched participant data consistently better than Ignore Groups; thus, these results support the thesis of this dissertation. The support is especially evident from the measure *%ErrorFreeSuccess*. Figures 37c and 37d show the scatter plots of *%ErrorFreeSuccess* by Consider Groups and Ignore Groups respectively compared to participant data, where each data point represents one of the 36 tasks. Without group information to guide exploration, the large number of selectable links (93 links in total) makes it less likely that the model will see and evaluate the correct link before it decides to stop and select the best link seen so far. Thus, while Consider Groups was able to select the correct link on its first attempt on the same tasks as participants did (upper right quadrant of Figure 37c), Ignore Groups failed to do the same on those tasks (upper left quadrant of Figure 37d).

Furthermore, the same model parameters used in CogTool-Explorer 1.1 for the multi-page layout and in CogTool-Explorer 1.2 for the half-flatten layout transferred to CogTool-Explorer 1.2 for the multi-group layout without worsening the fit between model and participants very much; the results from CogTool-Explorer 1.2 (Table 12: Consider Groups) are comparable to the results from CogTool-Explorer 1.2 (Table 11) and CogTool-Explorer 1.1 (Table 10). This suggests that the model parameters are not over fitted to a particular layout and do work in other layouts.

Table 12: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the multi-group layout. The better results are highlighted in bold.

	Correlation, R^2 (95% confidence interval)		%AAE	
	Consider Groups	**Ignore** Groups	**Consider** Groups	**Ignore** Groups
%Success	**0.67** (0.63, 0.71)	0.46 (0.39, 0.52)	**30.1%**	33.2%
%ErrorFreeSuccess	**0.72** (0.68, 0.76)	0.39 (0.32, 0.45)	**45.0%**	86.9%
LinkClicks	**0.33** (0.32, 0.33)	0.25 (0.25, 0.26)	**114%**	132%

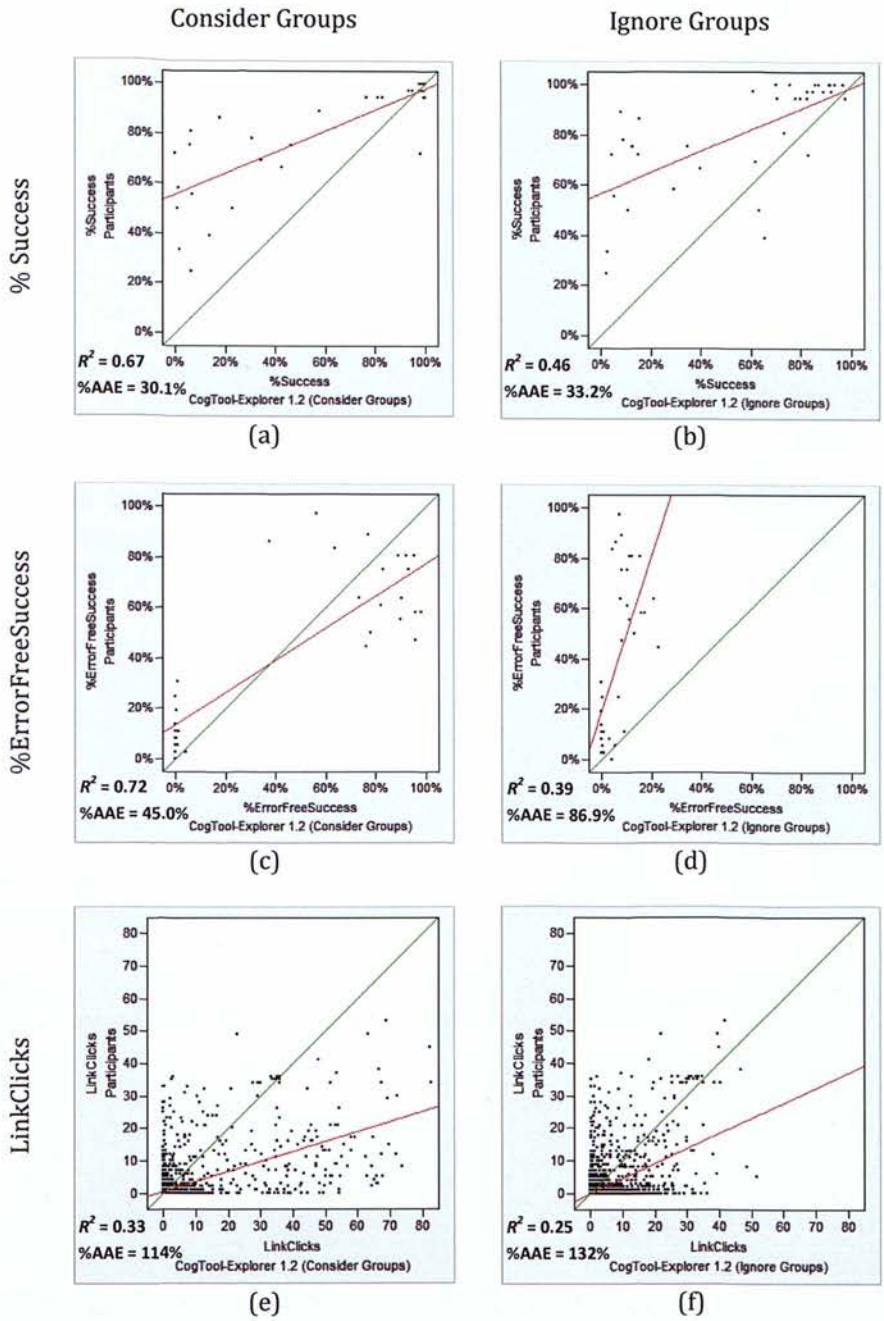


Figure 37: Consider Groups and Ignore Groups by CogTool-Explorer 1.2 compared to participant data in the multi-group layout. Each data point in (a) (b) (c) and (d) represents a task, and in (e) and (f) a link in a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

4.2.4 Comparison of CogTool-Explorer to AutoCWW's Predictions

The multi-page, half-flatten and multi-group layouts, tasks and participant data used in this dissertation were previously used in the research and development of AutoCWW (Blackmon et al., 2005) and its related research (for example Toldy, 2009), thus, it is both relevant and interesting to test how CogTool-Explorer compares to AutoCWW's predictions in these 36 tasks in each of the three layouts. There are two other prior related works that consider grouping (Table 1: Grouping), however, CoLiDeS is not an executable model and it's references to visual search and grouping lacks implementation details; and DOI-ACT assumes knowledge specific to the DOI tree UI layout and uses human ratings for category scent scores. Thus, I could not readily create and run these models to generate predictions.

As described in Section 2.6, AutoCWW predicts the mean number of link clicks that "users will make to accomplish a particular task on a specific webpage" (Blackmon et al., 2005, p. 31), that is, the mean number of link clicks in that webpage up to and including the click on the correct link. Blackmon et al. (2005) explained that *predicted mean total clicks* is "a measure of task difficulty", the more clicks to success for a task, the harder the task is. A low *predicted mean total clicks* would also suggest that the UI supports the successful exploration and completion of the task.

To compare AutoCWW's *predicted mean total clicks* to CogTool-Explorer, another task performance measure, *MeanClicksToSuccess*: the mean number of link selections in participant trials or model runs to accomplish each task, is computed from both participant data and CogTool-Explorer model runs[9]. Since AutoCWW's *predicted mean total clicks* is for a specific webpage, *MeanClicksToSuccess* by AutoCWW in the multi-page and half-flatten layouts can be derived by summing the *predicted mean total clicks* for the top-level page and the correct 2nd-level page. In the multi-group layout, since exploration takes place only on the top-level page, *MeanClicksToSuccess* by AutoCWW is simply the *predicted mean total clicks* for the top-level page.

Both Blackmon et al. (2005) and Toldy (2009) reported participant data and AutoCWW's predictions. However, for the multi-page layout, Toldy (2009) reported the mean total number of link clicks participants made on the top-level and 2nd-level pages (Table A-3 in Toldy, 2009), but did not separate successful from failure trials. Toldy (2009) also reported AutoCWW's *predicted mean total clicks*, but only for the 2nd-level page. For the half-flatten layout, the original experiment had 64 tasks (Cross-Validation Experiment in Blackmon et al., 2005) but results for individual tasks were not reported. However, this dissertation uses the subset of 36 tasks from those 64 tasks that are shared with the multi-page layout. For the multi-group layout, the participant data was collected from a new experiment that has not been previously reported in the literature, thus, no AutoCWW analysis has been reported for these 36 tasks in the multi-group layout. Therefore, in this dissertation, I tabulated *MeanClicksToSuccess* by participants from the original participant data

[9] Unsuccessful trials are excluded because some participants would click two or three links and then do nothing until time ran out whereas others continued to click (as does CogTool-Explorer). Also, AutoCWW's *predicted mean total clicks* is the number of link selections that users would make to accomplish a particular task on a specific webpage, which by definition, excludes unsuccessful trials.

files, and ran AutoCWW analyses for these 36 tasks on all three layouts. Section 2.6 gives a general walkthrough of how to use the AutoCWW website to analyze these tasks.

4.2.4.1 Multi-Page Layout

To setup the 36 tasks in AutoCWW, each task is submitted by first entering the task goal presented in the paragraph of text under the line "Find encyclopedia article about..." (Figure 16) as the goal statement. Next, the links in the top-level page and the links in the correct 2nd-level page for that task are entered into AutoCWW. As explained in Section 3.1, AutoCWW is designed to work with regions that have heading text, but the column of top-level links does not have heading text, therefore, the goal text is entered as the heading text for the top-level page. For the 2nd-level page, its corresponding top-level link is entered as the heading text of a single region of the 2nd-level links. Like in the two-column layout, AutoCWW is set to use the "General_Reading_up_to_1st_year_college (300 factors)" semantic space, and set to do the default full elaboration on the link texts because the original link texts are short. Unlike in the two-column layout, AutoCWW is set to do the default full elaboration on the heading texts because the 2nd-level page is a categorical grouping of the links, however, it should be noted that with only a single region in each of the top-level and 2nd-level pages, the header text will not affect AutoCWW's predictions (to be explained later in this section in the paragraph after Eq. 7), thus, the elaboration setting for header text will not matter but is still set accordingly for correctness. With these settings, AutoCWW will return the *predicted mean total clicks* for the top-level page and the 2nd-level page for each task.

Figures 38a and 38b respectively show the scatter plots of *MeanClicksToSuccess* by CogTool-Explorer 1.1 and by AutoCWW with the above settings, compared to participant data. CogTool-Explorer 1.1 had a R^2*MeanClicksToSuccess* of **0.72** and a %AAE of **42.4%**, whereas AutoCWW had a R^2*MeanClicksToSuccess* of **0.04** and a %AAE of **48.6%**. Given that AutoCWW was developed and tested on experiment websites and tasks similar to the multi-page layout, and that the settings described above for AutoCWW follow the examples in the AutoCWW research publications (best described in Kitajima et al., 2005, and in Blackmon et al., 2005) and in the AutoCWW tutorial[10], the poor match between AutoCWW and participant data on *MeanClicksToSuccess* is a surprise. AutoCWW's prediction formula for *predicted mean total clicks* (Eq. 7) points to two possible explanations for the poor performance by AutoCWW:

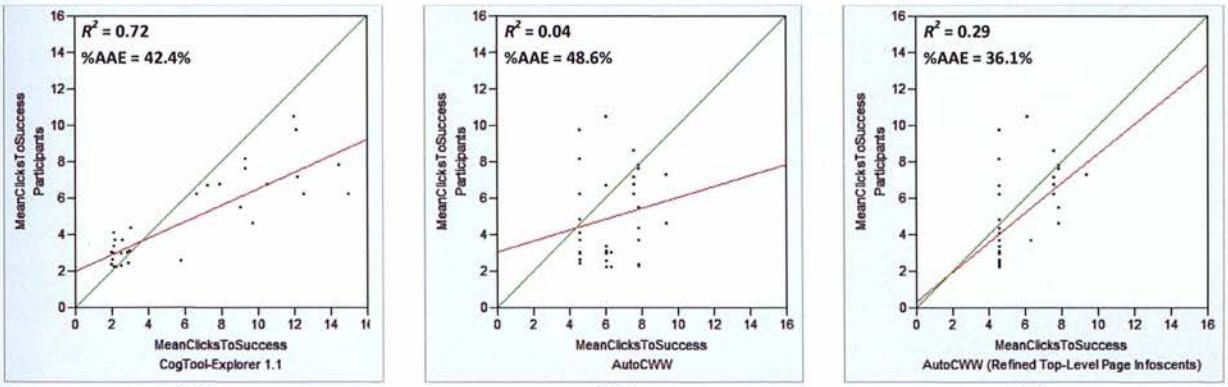
Mean Total Clicks = 2.292
+ 1.757 (if Link is unfamiliar)
+ 1.516 (if Link has a weak-scent)
+ 0.655 * (number of competing links nested under competing headings)
+ 0.0 * (number of competing links nested under the correct headings)
+ 0.0 * (number of competing headings)
Where Link refers to the correct link on the webpage

[Eq. 7]

First, the *predicted mean total clicks* is computed for each webpage. With two pages in the task and a minimum value or intercept of 2.292 in Eq. 7, AutoCWW predicts a minimum *MeanClicksToSuccess*

[10] Downloaded on October 17, 2010 from http://autocww.colorado.edu/

of 4.584 for a task in the multi-page layout that has no identified problems that hinder successful exploration. However, Figure 38b shows there are tasks where participants took less than an average of 4.5 clicks to success and Figure 38a shows CogTool-Explorer 1.1 matched participant data on those tasks. Second, Eq. 5 is explicit that the *number of competing links nested under the correct headings* does not add to the *predicted mean total clicks*. While this appears counter-intuitive, Eq. 5 was derived from a regression analysis of participant data from webpage layouts with multiple regions (Figures 2 and 17c) to identify the significant factors that hinders successful exploration, and the *number of competing links nested under the correct headings* was not one of them. Thus, in the case of the multi-page layout, where the top-level page and each of the 2nd-level pages have only one region in the page, competing links in the multi-page layout do not contribute to the *predicted mean total clicks*.



(a) (b) (c)

Figure 38: Results of *MeanClicksToSuccess* by CogTool-Explorer 1.1 and AutoCWW, compared to participant data in the multi-page layout. Each data point represents a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

In the multi-page layout, the computed infoscent scores for links in the top-level page differ between AutoCWW and CogTool-Explorer 1.1 due to different elaboration methods. As described in section 4.2.1.3, AutoCWW identifies all the words in the selected LSA corpus that occur at least 50 times in the corpus and have a minimum cosine of 0.5 with the link text vector, and appends these words to the link text before using LSA to compute the infoscent of the link. CogTool-Explorer 1.1 does the same but further appends the link texts from the 2nd-level links associated with this top-level link before using LSA to compute the infoscent of the top-level link. Thus, it will be interesting to see how AutoCWW will perform if it uses the same infoscent scores as CogTool-Explorer 1.1 uses for top-level links. To do this, in the setup of AutoCWW for the top-level page, the fully elaborated link text as per the method in CogTool-Explorer 1.1 is entered into AutoCWW, and AutoCWW is set to do no elaboration on the link text.

Figure 38c shows the scatter plot of *MeanClicksToSuccess* by AutoCWW in such a setup, which improved AutoCWW's R^2*MeanClicksToSuccess* from **0.04** to **0.29** and reduced %AAE from **48.6%** to **36.1%**. Comparing Figures 38c to 38b, this setup resulted in the data points representing tasks where participants took an average of 4 or fewer clicks to success, shifting towards the left

(AutoCWW now predicts these tasks will be more successful) and thus a better match to participant data, replicating a similar improvement as seen for CogTool-Explorer 1.0a in Section 4.2.1.3.

Finally, it is also interesting to contrast these *R²MeanClicksToSuccess* results by AutoCWW and CogTool-Explorer 1.1 with the *R²MeanClicksToSuccess* of 0.56 reported for DOI-ACT by Budiu and Pirolli (2007). Although DOI-ACT was developed and tested on a different UI layout and set of tasks, and therefore results are not directly comparable, it is notable that CogTool-Explorer 1.1 achieved a stronger correlation with participant data on this metric in this particular example.

4.2.4.2 Half-flatten Layout

The initial steps to set up the half-flatten layout in AutoCWW follows that for the multi-group layout in the previous section: each task is submitted by first entering the task goal presented in the paragraph of text under the line "Find encyclopedia article about..." (see Figure 29) as the goal statement. Next, the links in the top-level page and the links in the correct 2nd-level page for that task are entered into AutoCWW, and AutoCWW is set to use the "General_Reading_up_to_1st_year_college (300 factors)" semantic space.

There are two possible setups for the top-level page, following how in the previous section AutoCWW's predictions improved when using the same infoscent scores as CogTool-Explorer 1.1. The first setup is for AutoCWW to use the default full elaboration of link texts: identify all the words in the selected LSA corpus that occur at least 50 times in the corpus and have a minimum cosine of 0.5 with the link text vector, and appends these words to the link text before using LSA to compute the infoscent of the link. The second setup is to enter the fully elaborated link text as per the method in CogTool-Explorer 1.2, and set AutoCWW to do no elaboration on the link text.

Where this setup differs from the multi-page layout in the previous section is that now, in the half-flatten layout, there are two ways to model the 2nd-level pages in AutoCWW, similar to Consider Groups and Ignore Groups in CogTool-Explorer 1.2. The first way is as two regions of links, one region consisting of the 2nd-level links with the associated top-level link as the region's heading, and the other region consisting of the rest of the top-level links with the goal text as the region's heading (Sections 3.1 and 4.2.4.1 explain the use of the goal text as a region's heading). The second way is as a single region consisting of all the links in a 2nd-level page with the goal text as the region's heading. For both ways of modeling the 2nd-level pages, AutoCWW is set to do the default full elaboration for the 2nd-level links because the original link texts are short, and to do the default full elaboration for the heading texts because the region of 2nd-level links is a categorical grouping of those links. Although the region of top-level links is not a categorical grouping, the setting for elaboration of heading text is a global setting in AutoCWW and so priority is given to the region of 2nd-level links.

In combination, there are four possible configurations for AutoCWW and results from all four configurations are presented here. Figure 39 shows the scatter plots of *MeanClicksToSuccess* by CogTool-Explorer 1.2 and by the four configurations of AutoCWW. Compared to participant data, CogTool-Explorer 1.2 achieved *R²MeanClicksToSuccess* of **0.72** and **0.76**, and %AAE of **40.6%** and **25.3%**, respectively for Consider Groups (Figure 39a) and Ignore Groups (Figure 39b). For AutoCWW, the two configurations using the default elaboration method had *R²MeanClicksToSuccess*

of **0.17** and **0.08**, and %AAE of **46.1%** and **48.9%**, respectively for two regions (Figure 39c) and one region (Figure 39d), whereas when using the same infoscent scores as CogTool-Explorer 1.2 for the top-level page, AutoCWW had R^2*MeanClicksToSuccess* of **0.51** and **0.42**, and %AAE of **36.0%** and **37.1%**, respectively for two regions (Figure 39e) and one region (Figure 39f).

Interestingly, in contrast to the results by CogTool-Explorer 1.2 in the half-flatten layout where the participant data was matched no better by considering groups than ignoring groups, modeling the 2nd-level page as two regions in AutoCWW resulted in predictions that matched participant data better than modeling the 2nd-level page as a single region. However, as explained in the previous section (the paragraph after Eq. 7), when modeled as a single region, competing links in the page do not contribute to the *predicted mean total clicks*, thus, the better results when modeled as two regions may be a consequence of enabling competing links to influence the prediction. The results for AutoCWW also mirror the findings from the previous section: (1) AutoCWW predicts a minimum *MeanClicksToSuccess* of 4.584 for a task in the half-flatten layout, thus, is unable to match participant data on tasks where participants took less clicks to success, and (2) using the same infoscent scores as CogTool-Explorer 1.2 for the top-level page improved AutoCWW's predictions match to participant data.

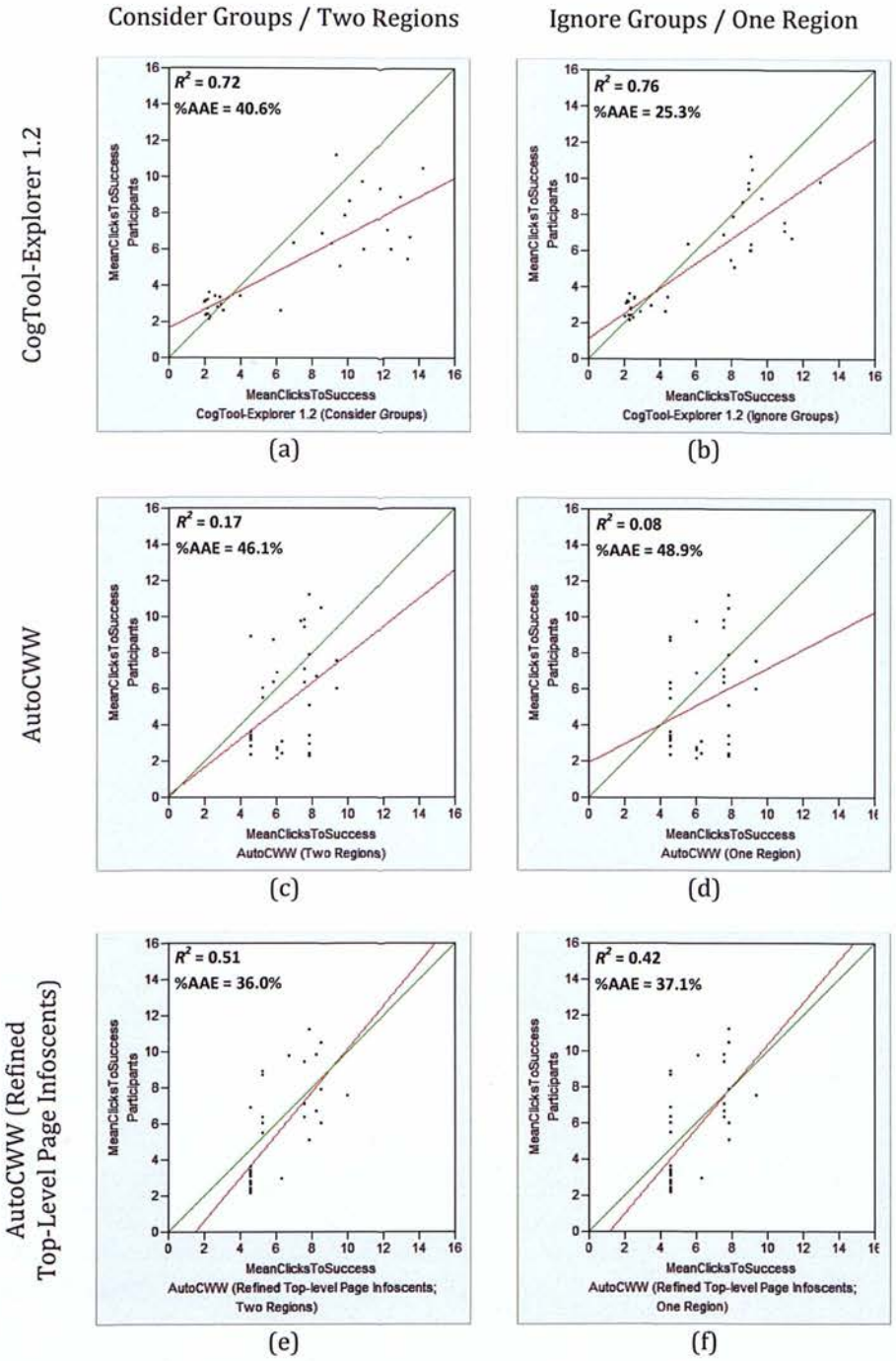


Figure 39: Results of *MeanClicksToSuccess* by CogTool-Explorer 1.2 and AutoCWW, compared to participant data in the half-flatten layout. Each data point represents a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

4.2.4.3 Multi-Group Layout

The current version of AutoCWW[11] was developed and tested on layouts identical to the multi-group layout, thus, the way to model the nine regions of links and the default settings to elaborate and compute the infoscent scores for links and regions are exactly stated in the examples from the AutoCWW research publications (best described in Kitajima et al., 2005, and in Blackmon et al., 2005) and in the AutoCWW tutorial[12] (Section 2.6 gives a general walkthrough). Each task is submitted by first entering the task goal presented in the paragraph of text under the line "Find encyclopedia article about..." (see Figure 34) as the goal statement. Next, the heading texts of the nine regions and the link texts of the 2nd-level links are entered into AutoCWW. AutoCWW is set to use the "General_Reading_up_to_1st_year_college (300 factors)" semantic space, and set to do the default full elaboration for the link texts because the original link texts are short, and to do the default full elaboration for the heading texts because the regions are categorical groupings of their member links. Both these default elaboration methods are described in Section 4.2.1.3.

Figure 40 shows the scatter plots of *MeanClicksToSuccess* by CogTool-Explorer 1.2 and by AutoCWW. Compared to participant data, R^2*MeanClicksToSuccess* by Consider Groups, Ignore Groups and AutoCWW were **0.63**, **0.28** and **0.45** respectively, and %AAE were **80.9%**, **81.3%** and **38.9%** respectively. The R^2*MeanClicksToSuccess* results add to the findings from the other measures presented in Table 12 and Figure 37, which is that for CogTool-Explorer 1.2 in the multi-group layout, Consider Groups matched participant data consistently better than Ignore Groups and supports the thesis of this dissertation.

However, the %AAE results by CogTool-Explorer 1.2 is double that of AutoCWW's. While Consider Groups by CogTool-Explorer 1.2 achieved a stronger correlation with participant data compared to AutoCWW, CogTool-Explorer 1.2 made more link selections before task success, as reflected in the shallower gradient of the best-fitting line (Figure 40a), compared to AutoCWW's predictions (Figure 40c). The better %AAE results by AutoCWW could be attributed to its formula for *predicted mean total clicks* [Eq. 7] being derived from a regression analysis of participant data from tasks performed in layouts identical to the multi-group layout. Section 4.2.1.7 offers another explanation for the poorer %AAE by CogTool-Explorer 1.2, which is that LSA tends to overestimate the infoscent between links and the task goal, and overestimation of infoscent would make more links look attractive to the model than they actually did to participants. This would make the model select links that participants did not, and select more links before going back from incorrect groups. Budiu and Pirolli (2007) made the same observation that their DOI-ACT model on average took more clicks than participants did to complete their tasks. Their explanation was that this indicates "the backtracking heuristics used by people is not fully captured by the (DOI-ACT) model."

---

[11] Accessed on October 17, 2010 at http://autocww.colorado.edu/~brownr/ACWW.php

[12] Downloaded on October 17, 2010 from http://autocww.colorado.edu/

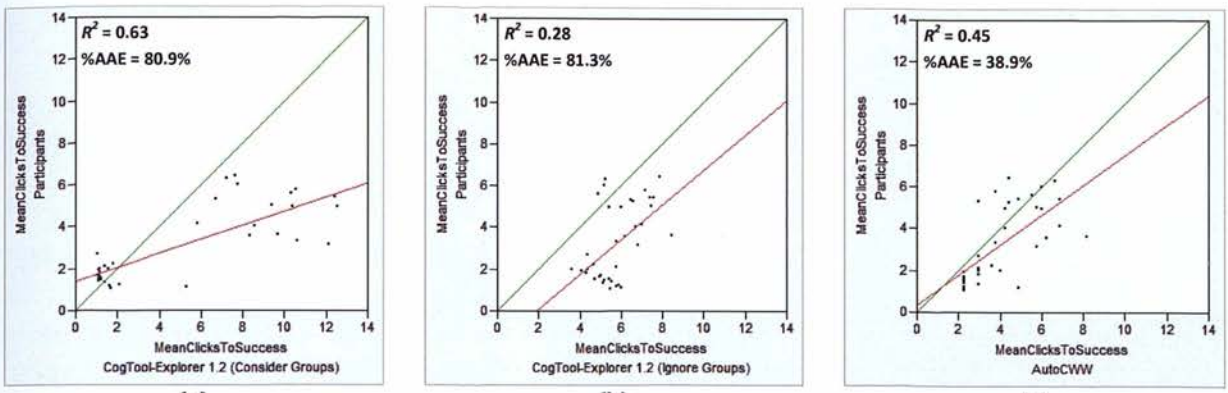


Figure 40: Results of *MeanClicksToSuccess* by CogTool-Explorer 1.2 and AutoCWW, compared to participant data in the multi-group layout. Each data point represents a task. If model behavior perfectly matched participant data, all data points will lie on the green diagonal line. The red line is the best fitting line for the data points.

4.2.5 Summary of Results

As explained in Section 4.2, the approach in this dissertation to make CogTool-Explorer consider grouping during exploration was to progressively test and modify CogTool-Explorer over three webpage layouts: multi-page, half-flatten and multi-group.

We started in Section 4.2.1 by running CogTool-Explorer 1.0 from the two-column layout (Section 4.1) on the multi-page layout, but the initial results were disappointing (Table 2). Inspecting model runs at a more detailed level, to find what might have led to the model's poor performance and what could be done to improve the match between model behavior and participant data, led to a series of model refinements: (a) changed how the infoscent of top-level links are computed to better reflect participant experience, (b) enabled reselection of previously selected links in a single model run because data showed that participants did so, refined the model's go-back behavior by (c) reducing the GoBackCost parameter, (d) adding a confidence mechanism and (e) updating the confidence the model has about the current page based on the outcome of link selections during exploration, and aligned the model's speed of execution to match that of human participants by (f) adding two perceptual-motor requirements that were missing from the UI mockup and (g) increasing the duration associated with the production that assesses the infoscent of a link. With all these refinements implemented, CogTool-Explorer 1.1 improved on all comparison metrics over CogTool-Explorer 1.0, as shown in Table 13 (results compiled from Tables 3 and 10).

Table 13: CogTool-Explorer 1.0 and 1.1 compared to participant data in the multi-page layout. The better results are highlighted in bold.

	Correlation, R^2 (95% confidence interval)		%AAE	
	CogTool-Explorer 1.0	CogTool-Explorer 1.1	CogTool-Explorer 1.0	CogTool-Explorer 1.1
%Success	0.28 (0.21, 0.35)	**0.75** (0.70, 0.79)	34.8%	**26.1%**
%ErrorFreeSuccess	0.44 (0.37, 0.51)	**0.82** (0.78, 0.85)	54.2%	**41.0%**
LinkClicks	0.25 (0.24, 0.25)	**0.38** (0.37, 0.39)	194%	**111%**
GoBacks	0.25 (0.23, 0.28)	**0.27** (0.24, 0.30)	90.3%	**84.4%**

In Section 4.2.2, we modified CogTool-Explorer 1.1 to enable the expression of group relationships in the device model, and to enable the user model to see and utilize the group relationships during exploration, resulting in CogTool-Explorer 1.2. We then ran two versions of CogTool-Explorer 1.2, one that considers groups and one that ignores groups, on the half-flatten layout. The results failed to show that considering groups during exploration made CogTool-Explorer 1.2 match participant data better, but the good news is the same model parameters used in CogTool-Explorer 1.1 for the multi-page layout transferred to CogTool-Explorer 1.2 for the half-flatten layout without worsening the fit between model and participants very much, as shown in Table 14 (results compiled from Tables 10, 11 and 12).

Table 14: CogTool-Explorer 1.1 compared to participant data in the multi-page layout, and CogTool-Explorer 1.2 compared to participant data in the half-flatten and multi-group layouts. The best results are highlighted in bold.

	Correlation, R^2 (95% confidence interval)				%AAE			
	CogTool-Explorer **1.1**	CogTool-Explorer **1.2**			CogTool-Explorer **1.1**	CogTool-Explorer **1.2**		
	Multi-Page	**Half-flatten**		**Multi-Group**	**Multi-Page**	**Half-flatten**		**Multi-Group**
		Consider Groups	**Ignore** Groups	**Consider** Groups		**Consider** Groups	**Ignore** Groups	**Consider** Groups
%Success	**0.75** (0.70, 0.79)	0.68 (0.64, 0.72)	0.66 (0.60, 0.71)	0.67 (0.63, 0.71)	**26.1%**	27.4%	26.4%	30.1%
%ErrorFreeSuccess	**0.82** (0.78, 0.85)	**0.82** (0.79, 0.84)	0.81 (0.78, 0.84)	0.72 (0.68, 0.76)	41.0%	41.7%	**33.6%**	45.0%
LinkClicks	0.38 (0.37, 0.39)	0.38 (0.37, 0.38)	**0.48** (0.47, 0.49)	0.33 (0.32, 0.33)	111%	135%	**105%**	114%

We then turned to the multi-group layout in Section 4.2.3, and ran the two versions of CogTool-Explorer 1.2, one that considers groups and one that ignores groups. This time, the results (Table 12) show that considering groups during exploration consistently made CogTool-Explorer 1.2 match participant data better, which supports the thesis of this dissertation. Furthermore, the same model parameters used in CogTool-Explorer 1.1 and in CogTool-Explorer 1.2 for the half-flatten layout worked for the multi-group layout without worsening the fit between model and participants very much (Table 14), which suggests that the model parameters are not over fitted to any particular layout.

Finally, in Section 4.2.4, we compared CogTool-Explorer to AutoCWW's predictions on *MeanClicksToSuccess* in all three layouts. Although CogTool-Explorer did not consistently do better than AutoCWW on %AAE, CogTool-Explorer consistently had higher R^2*MeanClicksToSuccess* with participant data compared to AutoCWW. These results suggest that CogTool-Explorer is at least as good and can be better than AutoCWW in differentiating between tasks at the extremes, that is, which tasks are sufficiently supported by the UI design and are mostly successful such that further design effort can be diverted to other areas, and which tasks and UI designs are less successful and thus in most need of further redesign effort.

4.3 Integration of CogTool-Explorer into CogTool

Section 3 presented the three research gaps in modeling goal-directed user exploration that are the focus of this dissertation: (1) consideration of layout position, (2) consideration of grouping and (3) implementation as a tool. Sections 4.1 and 4.2 presented the CogTool-Explorer model, to address both the first and second research gaps in consideration of layout position and grouping. This section presents design and implementation work done to integrate CogTool-Explorer 1.2 into CogTool, to address the third research gap in implementation as a tool.

CogTool-Explorer 1.0 and earlier versions were not integrated into CogTool, and required many file manipulation and software coding steps to setup and run the model, which is not easy for use by modeling researchers, and not practical for use by practitioners. With help from the CogTool software development team[13], this dissertation has integrated CogTool-Explorer 1.2 into CogTool, enabling a CogTool modeler to setup and run CogTool-Explorer models from within the direct-manipulation UI of CogTool. The light blue rectangles in region T of Figure 9 show the new or refined components that are part of this integration, and Sections 4.3.1 to 4.3.5 describe this integration in detail.

Furthermore, between January to March 2010, I conducted a series of design meetings with the CogTool software development team and Dr. Rachel Bellamy, a research collaborator from IBM's T. J. Watson Research Center. In these meetings, I proposed new designs and led the discussion, critique and redesign of various menus, dialogs and views that pertain to the integration of CogTool-Explorer into CogTool. Appendices A and B present the design outcomes from this series of meetings, and Sections 4.3.3, 4.3.4 and 4.3.7 make references to their relevant parts in these Appendices.

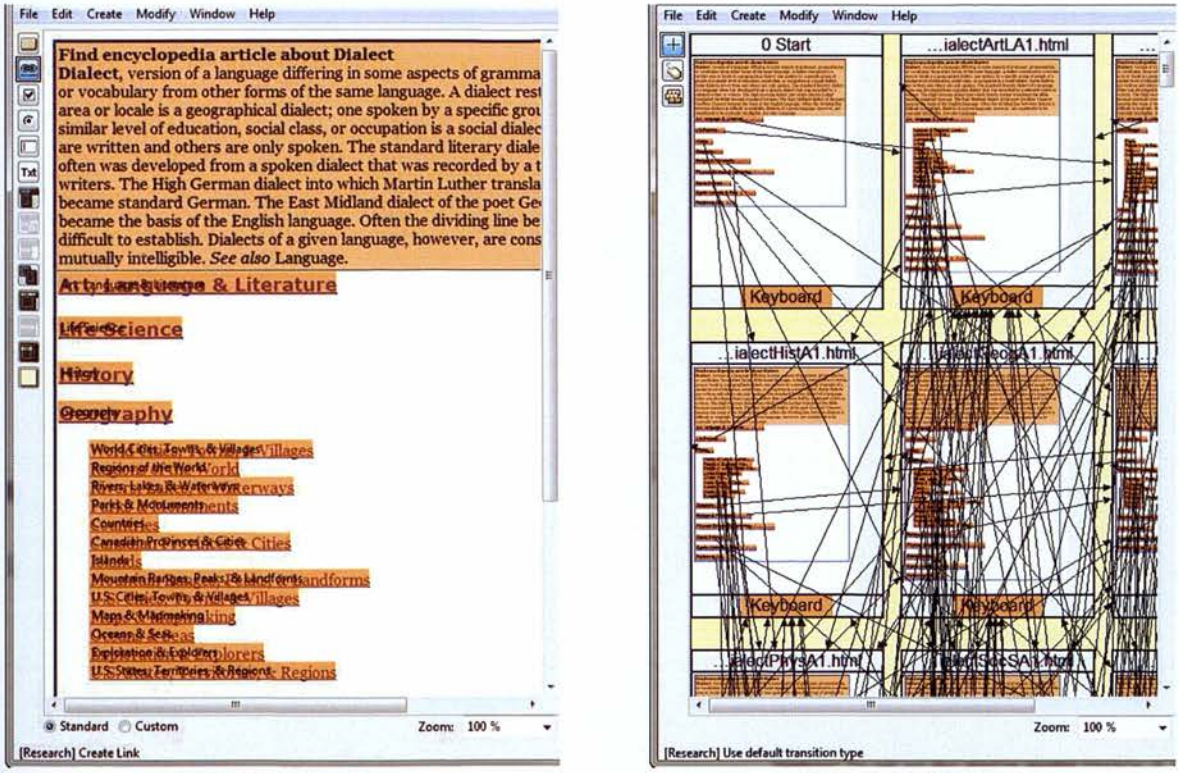
4.3.1 Import of Webpages from Websites

CogTool has from its inception supported the creation of UI mockups by hand (lower left of Figure 9). In CogTool's Frame Window (Figure 41a), a modeler can use the drawing tools located on the left of the window and drag and drop standard UI widgets such as buttons and links onto a frame. In CogTool's Design Window (Figure 41b), the modeler can create different frames and draw transitions from widgets to frames to indicate that certain actions on a widget, such as a mouse click, would result in the device model transiting to another frame, thus changing the display state of the UI.

However, creating a mockup of a reasonably sized UI by hand for the CogTool-Explorer model to freely explore in is a time-consuming task. CogTool-Explorer 1.0 made the first steps towards automating the creation of UI mockups by importing webpages from websites. To set up the device model (region D in Figure 9), the first step is to run a web-crawling component, implemented as a separate JAVA program, with the URL of the website and the depth of the website to crawl and capture. The program would render each webpage encountered in the crawl, extracts its layout (position, dimension and text label of each link in the webpage) and link information (the target URL of each link in the webpage) and write it out to a data file in a XML format that CogTool reads.

[13] Thanks to (in alphabetical order) Brett Harris, Dr. Michael Horowitz, Don Morrison and Ryan Myers.

The next step is to run CogTool and use its "Import from XML" function to read the data file and automatically create the UI mockup of the website, complete with all fames, links and transitions from links to their target frames (see Figure 41 for an example). The last step is to use CogTool's "Export to ACT-R" function to create an ACT-R device model of the UI mockup and write it out as a LISP source code file. The file contains the source code for a LISP Object that when installed into an ACT-R model, such as CogTool-Explorer 1.0, functions as the device model which the user model (region U of Figure 9) can interact with.



(a) Frame Window (b) Design Window

Figure 41: In CogTool's Frame Window, users can drag and drop standard UI widgets such as buttons and links from a palette of widgets onto a frame. A frame represents a display state in the UI. In CogTool's Design Window, users can create different frames and specify how interface actions on a widget, such as a mouse click, changes the display state of the UI, by drawing transitions from that widget in its frame to another frame.

Although CogTool-Explorer 1.0 made the creation of large UI mockups a lot less time-consuming, it was done in a separate program outside of CogTool and required further file manipuation steps by the modeler as described above. Beginning with CogTool-Explorer 1.0a, the JAVA code from the web-crawling component has been adapted and incorporated into CogTool. In CogTool-Explorer 1.2, the modeler can simply select the "Import Design from HTML" menu item, specify one or more starting URLs, depths and other parameters in a dialog box (Figure 42), and hit "OK" to import webpages from the Web directly into a UI mockup from within the CogTool application, thus, streamlining the automated import of webpages from websites (lower left of Figure 9).

Figure 42: Import HTML Pages dialog box

4.3.2 Specification of Groups

Section 4.2.2.1 described two sets of additions in CogTool-Explorer 1.2, one to the device model (region D in Figure 9) to express group relationships in the UI, and the other to the user model (region U in Figure 9) to see the group relationships now expressed in the device model and follow a group-based hierarchical exploration process. In this dissertation, group relationships are provided by the human modeler as input to CogTool-Explorer 1.2, thus, we added new functionality to CogTool's Frame Window to enable the modeler to specify and manipulate groups of widgets in the UI mockup (lower left of Figure 9). In a frame with widgets (for example Figure 41a), the modeler could select multiple widgets and issue the new "Group" command from either the menu or its keyboard shortcut. A group composed of the previously selected widgets would be created and automatically selected (Figure 43a). The properties panel on the right of the Frame Window would show the name of the group, which the modeler can edit, its Remote Label and Auxiliary Text (if any and also editable; described in Section 4.3.2.1) and a list of widgets that belong to this group. Clicking on a widget in the list, or directly on a widget that belongs to some group in the frame, will not only select the widget and highlight it with a gray border (which is standard CogTool behavior), but also highlight the group or groups that this widget belongs to with a red border (Figure 43b).

Although not needed for modeling the UI layouts in this dissertation, we have also implemented the ability to created nested groups. The modeler can select multiple widgets and/or groups, and group them just as described in the previous paragraph. Although CogTool-Explorer 1.2 did not encounter nested groups in this dissertation, the model would behave appropriately if there were nested groups: if the model is currently focused on a group and decided to next focus on a group nested in the current group, the model would update the exploration history with the current group and exploration would continue within the nested group, and when the model decides to go back from the nested group, the model would revert its focus to the parent group by reading its entry off the exploration history.

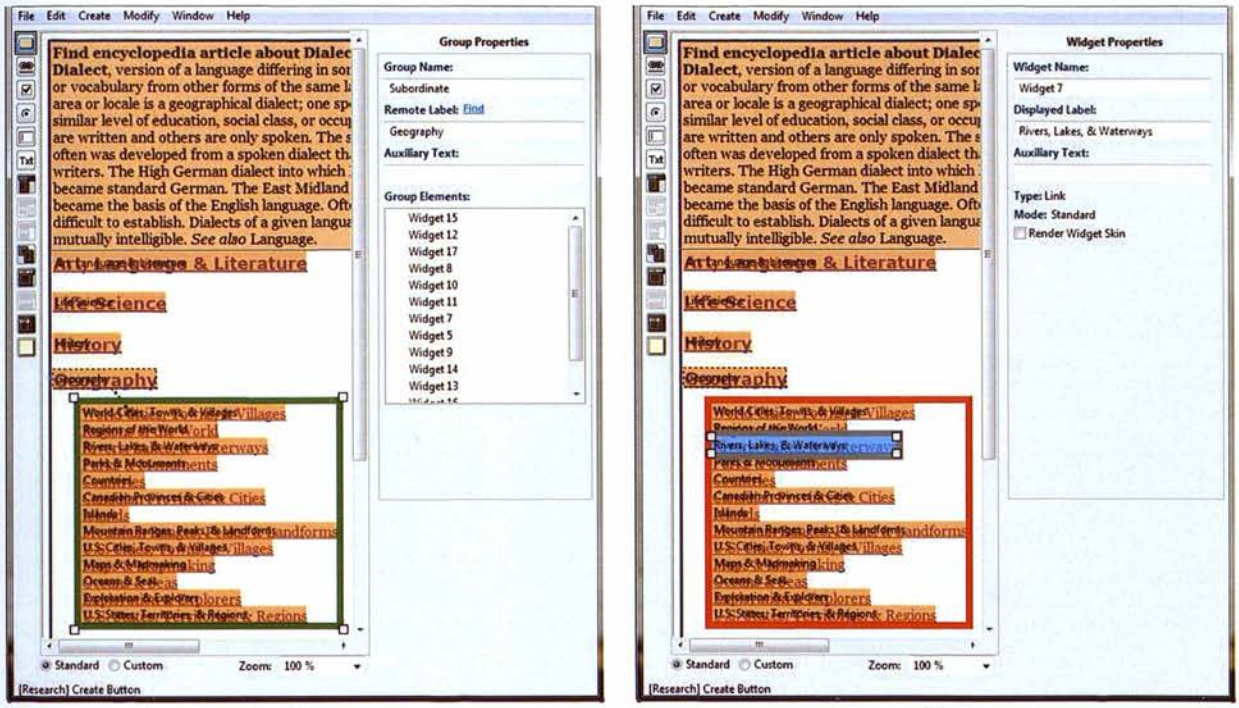


Figure 43: To specify existing widgets form a group, the practitioner would multiple select these widgets and then issue the "Group" command from either the "Modify" menu or its keyboard shortcut. This will create a group composed of the previously selected widgets and the newly created group will be automatically selected with the group's bounding box highlighted in green (Figure a). Selecting a widget that is a member of a group will highlight the widget with a gray border (which is standard CogTool behavior) and also highlight the group with a red border (Figure b).

Another situation that was not encountered in this dissertation, but which we have also implemented, is the ability to specify that a widget or group (referred to in common as elements) belongs to more than one group. When the modeler is selecting multiple elements for inclusion in a new group, he or she can select an element that is previously included in the creation of some other group. In the device model (region D in Figure 9) of CogTool-Explorer 1.2, if an element belongs to two groups, two visual-location objects would be created for this element. These two visual-location objects would be identical in every way except in the value of their *member-of-group* slot, where one visual-location object would reference the first group, and the other visual-location object would reference the second group. In the user model (region U in Figure 9) of CogTool-Explorer 1.2, when the model makes a request to the ACT-R vision module, either visual-location objects could match. This is our first approximation at modeling how CogTool-Explorer 1.2 would perceive that an element belongs to more than one group.

4.3.2.1 Textual Cues of Widgets, Groups and Remote Labels

Each widget in a CogTool UI mockup can have a Displayed Label, for example, the selected link in Figure 43b has a Displayed Label "Rivers, Lakes, & Waterways", which is rendered as part of the widget in the drawing area of the Frame Window, and is shown in the properties panel on the right

of the Frame Window in an editable text field. In the device model, the Displayed Label becomes the value of the *value* slot in the ACT-R visual object representing the widget. In CogTool-Explorer 1.1 and earlier, when the model shifts its point of visual attention to read a link, the model takes the text in the *value* slot and the text from the task goal to compute the infoscent of the link with respect to the goal.

To enable greater flexibility in the construction of the text string that is used to compute the infoscent of a widget, in CogTool-Explorer 1.2, we added an Auxiliary Text to each widget in a CogTool UI mockup. A widget's Auxiliary Text is also shown in the properties panel on the right of the Frame Window in an editable text field, but is not rendered as part of the widget in the drawing area of the Frame Window. Now, in the device model, the Displayed Label becomes the value of a new *display-label* slot (as well as the value of the original *value* slot) in the visual object representing the widget. In addition, the visual object has a new *textual-cue* slot whose value is the concatenation of the texts in the Displayed Label, the Auxiliary Text and the *textual-cue* of its Remote Label (to be described later in this section). In CogTool-Explorer 1.2, when the model shifts its point of visual attention to read a link, the model takes the text in the *textual-cue* slot and the text from the task goal to compute the infoscent of the link with respect to the goal. The Auxiliary Text and the *textual-cue* slot enable the manual entry of additional text that elaborates the Displayed Label to better capture what the widget might mean to a user for the model, without editing the text of the Displayed Label. John, Blackmon, Polson, Fennell and Teo (2009) used the Auxiliary Text to elaborate buttons whose display label is an abbreviation, for example, "VNAV" on the Control Display Unit in the Boeing 777's cockpit means "Vertical Navigation" and every Boeing 777 pilot knows that, so those words were entered as Auxiliary Text.

Unlike a widget, a group in a CogTool UI mockup does not have a Displayed Label because it is the widgets in the group and its nested groups that determine what the group looks like. A group can have an Auxiliary Text and/or a Remote Label, and the group's *textual-cue* slot in the device model recursively constructs what the group looks like to the model: the *textual-cue* of a group is the concatenation of its Auxiliary text, the *textual-cue* of its Remote Label and the *textual-cue*s of its constituent widgets and/or nested groups.

Finally, in CogTool-Explorer 1.2, groups and some widget types like buttons can have a Remote Label. For example, in devices with soft buttons, that is, physical buttons that perform different functions depending on what is displayed on the device's display located away from the button, like the aforementioned Boeing Control Display Unit or cellular phones that use the numeric keypad instead of touch screens; the button's Remote Label enables the accurate placement of the button's label at a location other than on the button itself in the UI mockup. The Remote Label appears in the UI mockup like the Displayed Label and is included in the widget's *textual-cue* slot in the device model. In this dissertation, groups with Remote Labels are used in the mockups for the half-flatten and multi-group layouts, to model the groups of 2nd-level links with their associated top-level link (in the half-flatten layout) or text heading (in the multi-group layout). Figure 44 shows the multi-group layout with the Performing Arts group selected. The properties panel on the right of the Frame Window shows that the group has a Remote Label "Performing Arts" (which is the text widget located above the group) and no Auxiliary Text. Clicking on "Find" or the "Performing Arts"

text widget located above the group will select the Remote Label. Figure 45 shows the Remote Label "Performing Arts" selected and its owner, the Performing Arts group, highlighted by gray and red bounding boxes respectively. The properties panel shows that the Remote Label has "Performing Arts" as its Displayed Label and "arts art artistic artists painters sculpture sculptures architecture" entered as its Auxiliary Text.

As described previously in this section, the Performing Arts group's *textual-cue* slot is the concatenation of its Auxiliary text (which is blank in this example), the *textual-cue* of its Remote Label (which is the concatenation of its Displayed Text "Performing Arts" and Auxiliary Text "arts art artistic artists painters sculpture sculptures architecture") and the *textual-cues* of its constituent widgets and/or nested groups. The constituent widgets of the Performing Arts group are six links with their respective Displayed Texts but all have neither Auxiliary Texts nor Remote Labels, thus, the *textual-cue* slot of the visual object representing the Performing Arts group has the value:

> "Performing Arts arts art artistic artists painters sculpture sculptures architecture Theater Musicians & Composers Cinema, Television, & Broadcasting Music Dance Musical Instruments"

The text entered for the group's Remote Label's Auxiliary Text, "arts art artistic artists painters sculpture sculptures architecture", is the standard elaboration (as described in Section 4.2.1.3) of the text "Performing Arts", with words from the "General_Reading_up_to_1st_year_college (300 factors)" LSA corpus that occur at least 50 times in the corpus and have a minimum cosine of 0.5 with the "Performing Arts" text vector. Thus, the above *textual-cue* for the Performing Arts group is the same as the text used for computing the infoscent of the top-level link "Performing Arts", as described in Section 4.2.1.3.

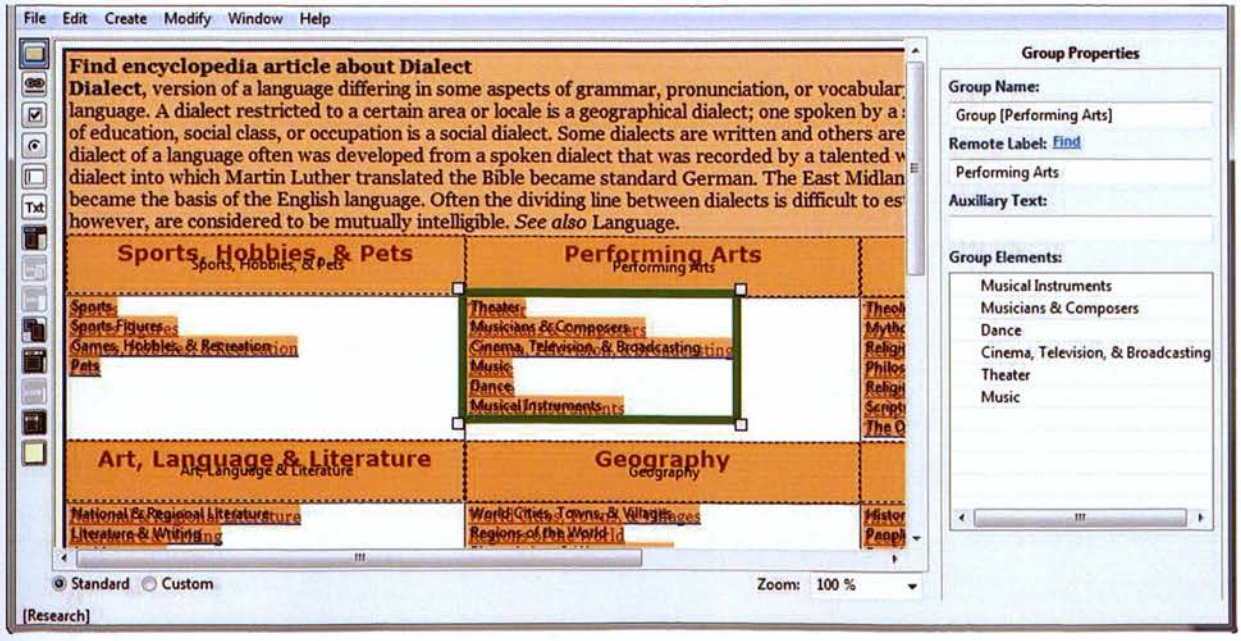


Figure 44: The multi-group layout with the Performing Arts group selected, highlighted by a green bounding box.

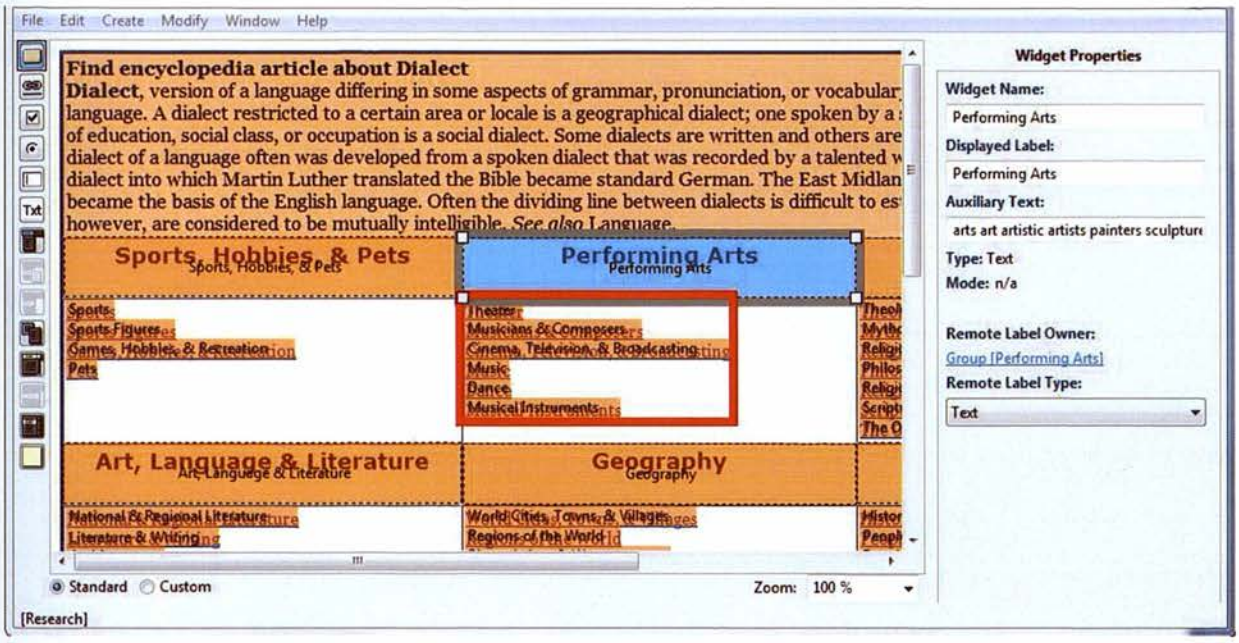


Figure 45: The multi-group layout with the Remote Label "Performing Arts" selected, highlighted by a gray bounding box. Selecting a Remote Label will also highlight its owner, in this example the Performing Arts group, with a red bounding box.

4.3.3 Retrieval of Information Scent Scores

CogTool-Explorer 1.0 and earlier versions required many file manipulation and software coding steps to setup the model. Prior to integration with CogTool, the modeler had to manually start up a LISP session, load the ACT-R architecture, and load three LISP files, one with the source code for the user model (region U in Figure 9), the second with the source code for the device model (region D in Figure 9), and the third with the infoscent scores of all possible pairs of goal and link texts that the user model could encounter in the device model. These infoscent scores were generated by manually submitting the goal and link texts to AutoCWW and then copying, from the Excel files that AutoCWW returns, the LSA cosine value for each goal-link pair into the third LISP file.

In CogTool-Explorer 1.2, the manual process of submitting texts to AutoCWW, retrieving the LSA cosine values and creating the third LISP file, which was laborious and error prone, has been automated (upper left of Figure 9). After the UI mockup has been imported or created, the modeler can select the "Generate Dictionary" menu item from a data cell at the intersection of a Task Description (the goal text) and a UI mockup in CogTool's Project Window (Figure 46). This will bring up a dialog box to specify the source of the infoscent scores to use and its related parameters. For example, if LSA is selected, one can select the appropriate LSA semantic space (corpus) to best match the demographics of the type of users the modeler wants predictions for. The modeler can change the standard elaboration settings from the default values of 50 and 0.5 (as described in Section 4.2.1.3) in the LSA URL field if desired. Clicking on "OK" will start a separate processing thread that, for each widget or group in the UI mockup, pairs its Displayed Label and *textual-cue* with the goal text and then submits the two pairs to AutoCWW to compute their LSA cosines.

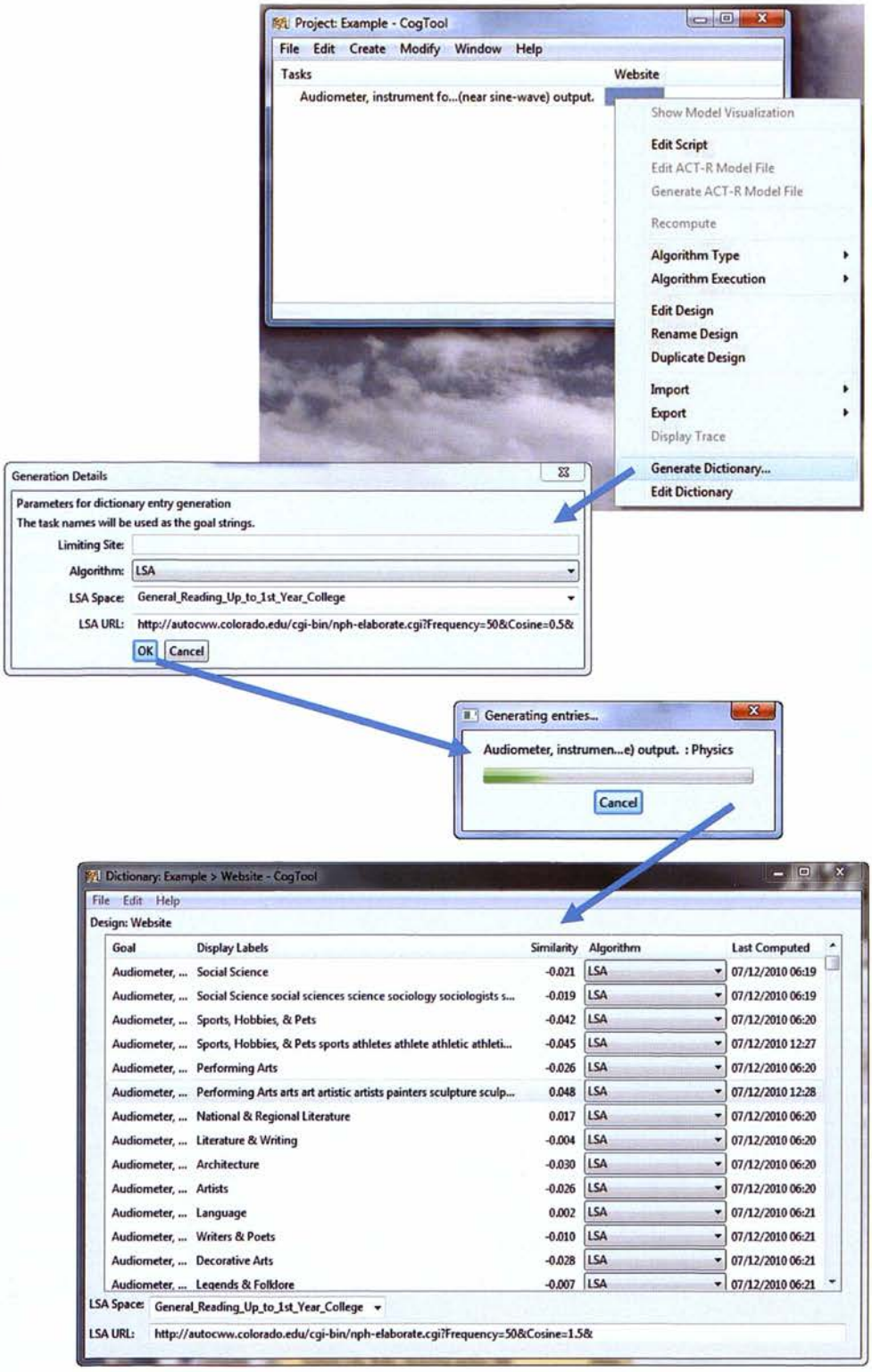


Figure 46: Select "Generate Dictionary" from the cell at the intersection of a Task Description and a UI mockup to generate infoscent scores for every Displayed Label and textual-cue in the UI design. The modeler can inspect and modify parameters and infoscent scores using the Dictionary Viewer window (bottom of Figure). For example, the highlighted row in the Viewer has its cosine parameter changed from 0.5 to 1.5 to disable further elaboration.

After all the LSA cosine values have been retrieved from AutoCWW, the values are shown in the Dictionary Viewer, which the modeler can also get to by selecting the "Edit Dictionary" menu item. The Dictionary Viewer provides a way for the modeler to inspect the infoscent scores, change the parameters for computing the scores, and even change to a different source of infoscent scores, including manual entry of scores by the modeler. For example in Figure 46, the selected row in the Dictionary Viewer is for the *textual-cue* of the Performing Arts group. Since the *textual-cue* of the group is already its elaboration as described in Section 4.3.2.1, it should not further undergo the standard elaboration performed by AutoCWW, thus, the LSA URL field for the *textual-cue* of the group is edited to use a minimum cosine value of 1.5, which has the effect of disabling the standard elaboration performed by AutoCWW.

Appendix A-4 presents a design of how the "Generate Dictionary" step could be further integrated into the setup of model runs

4.3.4 Setup of Model Runs

CogTool-Explorer 1.0 and earlier versions required many file manipulation and software coding steps to run the model. Prior to integration with CogTool, the modeler had to edit LISP source code files to specify the information required for each task: the text of the task goal, the name of the frame to start exploration from, and the name of the frame which indicates a successful model run. After entering the information for one or more tasks, the modeler would edit the LISP source code to specify the number of model runs to do for each task and the duration to run the model for before the run is considered a failure. After all these edits, the modeler had to manually start up a LISP session, load the ACT-R architecture and the three LISP files for the user model (region U in Figure 9), the device model (region D in Figure 9) and the infoscent scores into the LISP session, and then issue a command on the command line to start the model runs.

In CogTool-Explorer 1.2, the manual editing of LISP source code files, which was laborious and error prone, has been replaced with a new dialog box inside CogTool (upper right of Figure 9). After the UI mockup has been imported or created, and the infoscent scores retrieved, the modeler can setup model runs by selecting the "Recompute Script"[14] menu item from a cell at the intersection of a Task Description (the goal text) and a UI mockup in CogTool's Project Window (Figure 47). This will bring up a dialog box to specify the number of model runs to do, the value for the model's k parameter (defaults to 600; see Section 4.1.5), the frame to start exploration from, and the frame or frames that indicates a successful model run. Clicking on "OK" will automatically start a LISP session, load the ACT-R architecture and the three LISP files for the user model, the device model and the infoscent scores into the LISP session, and issue a command to the LISP session to run the model for the specified number of runs. On completion, each model run will be listed in CogTool's Project Window.

Appendices A-1 to A-5 present a design for the setup of CogTool-Explorer model runs.

---

[14] "Recompute Script" is a menu label carried over from CogTool. A more appropriate and descriptive menu label for CogTool-Explorer is presented in Appendix A-1.

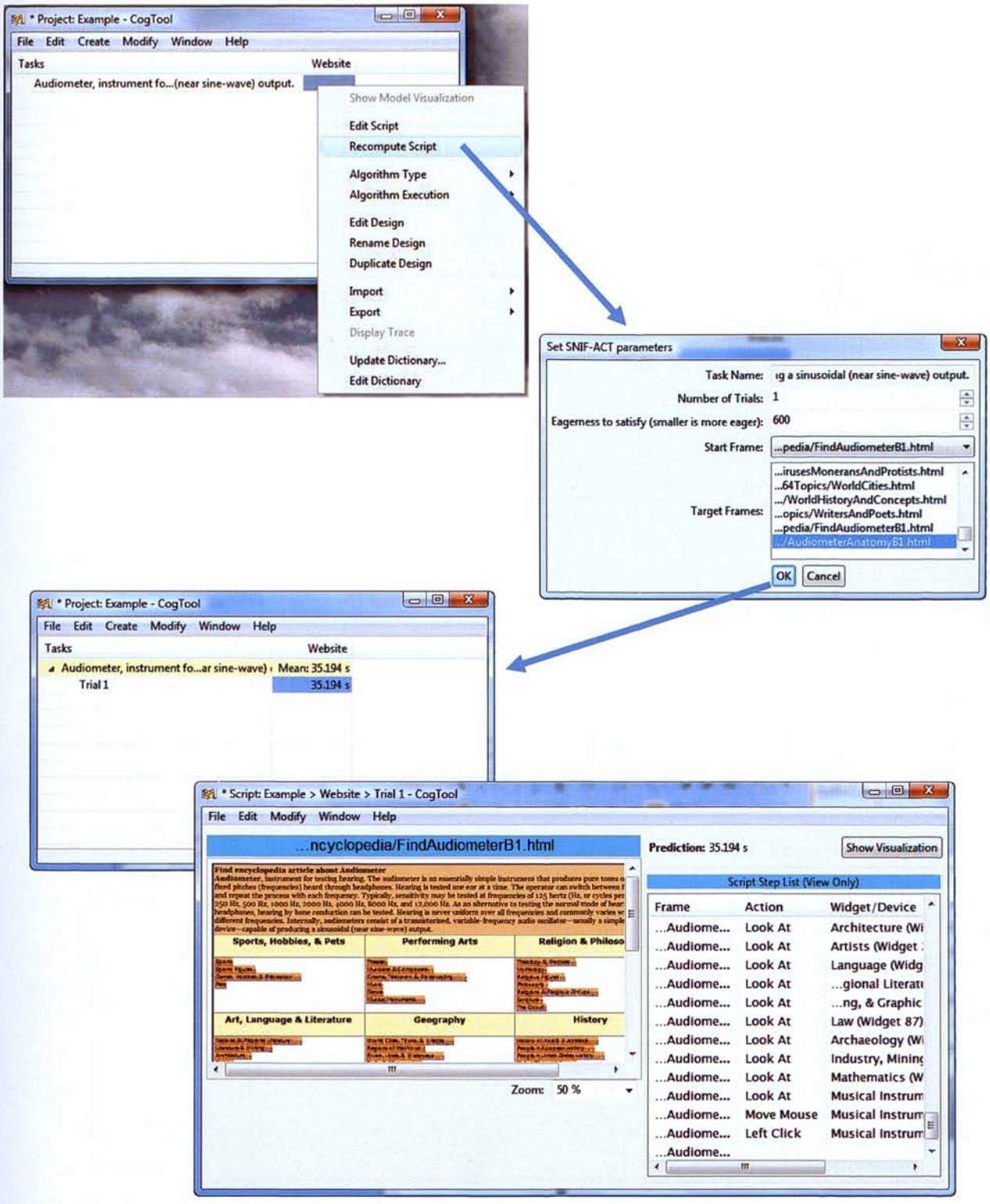


Figure 47: Select "Recompute Script" from the cell at the intersection of a Task Description and a UI mockup to bring up the dialog box to set up model runs, including the number of runs to do, parameter values such as k ("Eagerness"), the frame to start exploration from, and the frame or frames that indicate a successful model run. Each model run will be listed in CogTool's Project Window and can be further inspected in CogTool's Script Step Window.

4.3.5 Presentation of Model Runs

In all versions of CogTool-Explorer, the model productions associated with the two or three possible actions the model can take (read another link, select best link so far, or go back) will invoke additional LISP code each time the model takes an action during a model run, to write a record of the action taken to an external log file. Like most ACT-R models that use default ACT-R architecture settings, CogTool-Explorer also outputs a trace of production firings during a model run. Although it is possible to parse the ACT-R trace to extract the sequence of model actions, it is easier to record the actions in the log file as the model takes them. The modeler would then load the log file into a statistical software package like JMP or spreadsheet software like EXCEL to analyze, graph and compare the model runs to human data. In this dissertation, I wrote additional Python scripts (Appendix D) to further process the log file to tabulate the various task performance measures over multiple model runs for each task, before using JMP and EXCEL to analyze, graph and compare the model runs to participant data.

In CogTool-Explorer 1.0 and earlier versions, prior to integration with CogTool, these model runs do not appear inside CogTool, thus, the modeler is unable to utilize existing CogTool's features to inspect CogTool-Explorer model runs. In CogTool-Explorer 1.2, model runs are set up and executed from inside CogTool as described in the previous section. CogTool already knows how to parse an ACT-R trace to generate visualizations of a model run, thus, we made each CogTool-Explorer model run appear in CogTool's Project Window so that the modeler can inspect each model run in CogTool's Script Step Window (bottom of Figure 47) to see exactly that the model did in that particular run.

4.3.6 Another Usage Example of the Modeling Tool

The integration of the CogTool-Explorer model into CogTool, described in Sections 4.3.1 to 4.3.5, made it easy to set up and run CogTool-Explorer models on the text-based webpage layouts used in this dissertation. Over the course of this integration work, CogTool-Explorer was also used to model a completely different task in a radically different UI layout (John et al., 2009), that of entering an aircraft's landing speed using the Control Display Unit in a Boeing 777's cockpit (Figure 48). John et al. successfully used CogTool-Explorer to investigate what new theory, cognitive mechanisms and domain knowledge may be necessary to include in a general model of flight desk tasks, and at the same time demonstrated the use of CogTool-Explorer to model other tasks in a UI layout different from the text-based webpage layouts used in this dissertation.

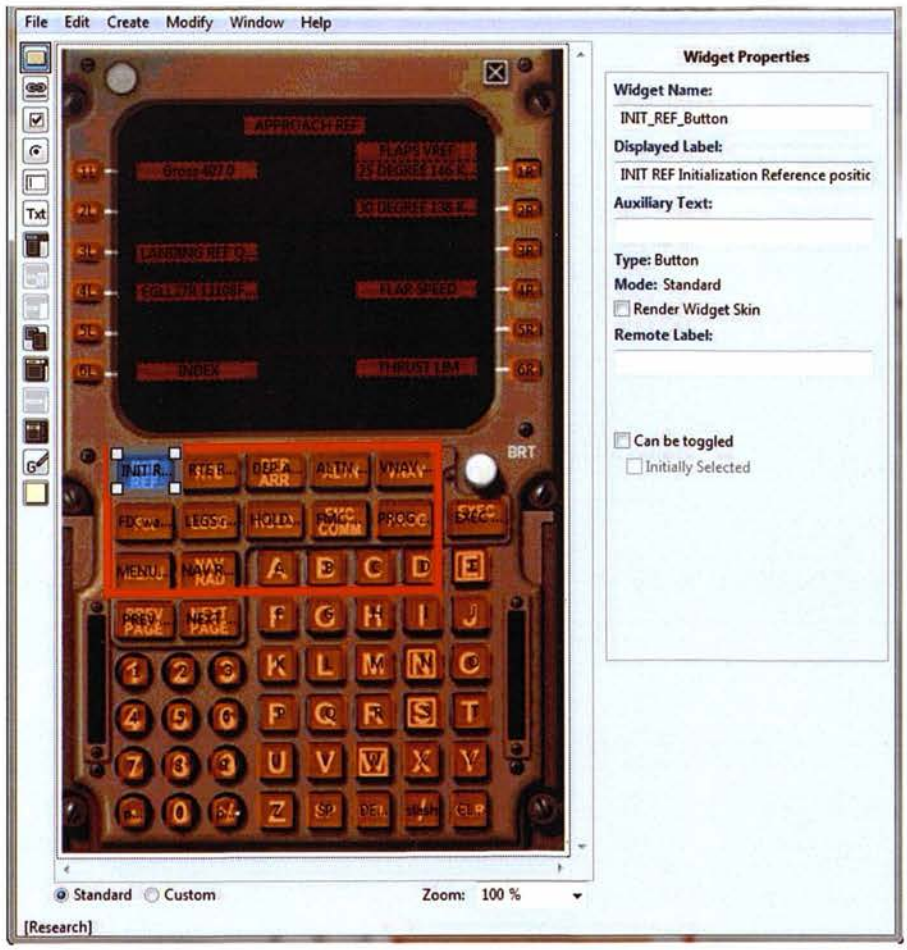


Figure 48: Mockup of the Control Display Unit (CDU) of a Boeing 777's cockpit. An image of the CDU was used as the background image of this frame. Button widgets were created over the parts of the image where a real button would be and text label widgets were created over the parts of the image where textual information would be displayed in the CDU's display. Note that the text string "Initialization Reference position", which describes the Displayed Label "INIT REF", was entered as part of the Displayed label before the Auxiliary Text field was implemented in CogTool-Explorer. Here, the "INIT REF" button is selected (highlighted with a gray border, which is standard CogTool behavior) and the Mode Keys group that the "INIT REF" button belongs to is highlighted with a red border (which is new in CogTool-Explorer).

The main contribution of the John et al. 2009 paper was to demonstrate how fast and easy it was to test out different models using CogTool-Explorer (circa 2009). In fact, the full title of the work, as presented at the conference, was "*Rapid Theory Prototyping: An Example of an Aviation Task OR How I explored five theories before breakfast.*" Table 15 adapts Table 1 in John et al. (2009) to illustrate this point (please refer to John et al., 2009, for details of the domain, task, interface and success metrics). The time it took to create each new model, ready to run, is shown in minutes in the third column.

Model Iteration	Manipulation in CogTool-Explorer to crete a new model	Time to make change (minutes)	% Success
i1	Original model		0
i2	Used CDU_skills corpus instead of 1st_year_College	1	0
i3	Elaborated goal & labels using flight manuals Put labels on the soft keys, not on the screen	180	0
i4	Added intermediate frames with regions and paths for recovery that remove an incorrect region after selection	60	3
i5	Removed button from frame after it was hit	1	4
i6	Added frames to visually check the alternative selection and to recover from an error	5	4
i7	Combined previous 2 changes	2	92

Table 15: Seven iterations of a model, the changes made in CogTool-Explorer to produce each iteration, the time it took to make those changes, and the %Success of the resulting model. (Adapted from Table 1, John et. al., 2009)

Some times are very short, 5 minutes or less, reflecting features in CogTool-Explorer that were already implemented in early 2009. For example, iteration i2 changed the retrieval of information scent scores (Section 4.3.3) from the default "General_Reading_up_to_1st_year_college (300 factors)" semantic space to a new semantic space built from instruction manuals that pilots are likely to have read, so that the information scent sores would reflect the knowledge of airline pilots rather than college students. Switching semantic spaces in CogTool-Explorer involved simply selecting another option in a drop-down list (see middle of Figure 46).

Other changes were more time-consuming, providing design guidance for improving the expressibility and efficiency of CogTool-Explorer as a modeling tool. We have made these changes in the version of CogTool-Explorer described in this dissertation. For example, i3 in Table 15 took 180 minutes to elaborate goals and labels and use a work-around to express soft-key labels that appear on the screen instead of on the buttons themselves. While groups of widgets could be expressed in 2009, not only did their own labels have to be typed in by hand, but all the labels and elaborations of the widgets included in the group had to be entered into the group label as well. This was not only time consuming but confusing. Now additional elaborations can be typed into the Auxiliary Text field and the textual cues for groups are automatically concatenated from the contents of the group (Section 4.3.2.1). Furthermore, the workaround in i3 to express the labels of soft-keys was replaced and simplified by the Remote Label feature (Section 4.3.2.1). Creating i4 (Table 15) took 60 minutes because the hierarchical visual search described in Section 4.2.3.1 was not yet implemented. In 2009, we used large button widgets to simulate groups and made each of these widgets transit to another frame that contained only the individual button widgets that belong to that group, thus simulating the group-based exploration process. This step would now come for free when specifying groups of widgets (Section 4.3.2), eliminating this entire hour's work. Thus, creating the models in John et al. (2009) would take substantially less time in the current CogTool-Explorer than it did in 2009, which itself was touted as extremely fast and easy.

4.3.7 Further Design Iteration of Modeling Tool User-Interface

While CogTool's Project Window provides a way for the modeler to manipulate CogTool-Explorer model runs, and does it in a way that is consistent with KLM models, the current Project Window does not show the settings used to generate a set of CogTool-Explorer model runs. Furthermore, while CogTool's Script Step Window and other existing CogTool visualizations of model runs provide a way for the modeler to inspect CogTool-Explorer model runs, they were designed to visualize a single model run or to compare between two model runs at a time, but not designed to visualize and compare the large sets of model runs (as many as 900 model runs per task for convergence in this dissertation) that CogTool-Explorer could generate. Appendices A-5 to A-18 presents a design for the Project Window, to show a set of CogTool-Explorer model runs and the settings used to generate it. Appendices B-1 to B-6 presents a design for visualizing a set of CogTool-Explorer model runs and for comparing a set of CogTool-Explorer model runs to another model run or a KLM model, for example, a KLM model of skilled execution in the same task.

Although the UI for CogTool-Explorer will inevitably need to evolve as more people use it, just as the UI for CogTool has changed over the years, the work already done to integrate CogTool-Explorer into CogTool is the first step that has enabled a few researchers other than myself to use CogTool-Explorer. The design work presented in Appendices A and B builds on our experiences so far, and represents a second design iteration toward making CogTool-Explorer into a practical tool for UI design and evaluation.

5 Contributions, Limitations and Future Work

Modeling research and tool development for predicting goal-directed user exploration in HCI has a rich history spanning more than two decades before CogTool-Explorer. Building on these prior works, this dissertation identified and focused on two modeling research gaps that a model of goal-directed user exploration must take into consideration to make accurate predictions of likely user behavior: the layout position and the grouping of selectable options in the UI layout. This dissertation also identified the lack of models that were further implemented as modeling tools, for practitioners to use alongside user testing and other HCI methods when designing and evaluating UIs. The thesis of this dissertation is:

> *A modeling tool for goal-directed user exploration of user-interfaces, that considers the information scent, the visual search process, and the layout and grouping of options in the user-interface in concert, can make more accurate predictions of user exploration compared to tools that do not.*

5.1 Contributions

The key contribution of this dissertation to the field of HCI is the new CogTool-Explorer model of goal-directed user exploration that considers the layout position and the grouping of options in the UI in concert with a serial evaluation visual search process and information scent. The numerous tests in this dissertation show that predictions from CogTool-Explorer match participant data better than alternative models that do not consider layout position and grouping in the UI layout.

This dissertation also contributes to the field of HCI an enhancement to an existing modeling tool, CogTool. CogTool has great success in modeling skilled user task performance (John, 2010). By integrating the CogTool-Explorer model into CogTool and extending the existing CogTool with new menu options, dialogs and supporting modules, it is now possible to model and generate predictions of both skilled execution and novice exploration by users in a HCI task, all from the direct-manipulation UI of CogTool (Section 4.3).

While the above performance and utility of CogTool-Explorer is a step forward for HCI research, there are limitations to this dissertation work. The rest of this section will discuss these limitations and motivate future work.

5.2 Verification at the Mouse-Click Level

A limitation of this dissertation is that CogTool-Explorer 1.2 is currently verified only at the mouse-click level. Since layout and grouping in the UI are the main contributions of CogTool-Explorer 1.2, mouse-clicks are an indirect measure of the effects of these UI design elements. Unfortunately, the AutoCWW experiments lacked eye-tracking data, which would be a more direct measure of the location and sequence of eye fixations as participants perform the task, and can be used to develop and test CogTool-Explorer. In particular, as mentioned in Sections 4.2.2.4 and 4.2.3.3, the availability of eye-tracking data would have make it possible to compare between the model's and participants' go-back behavior from groups in the half-flatten and multi-group layouts, because go-back actions in these layouts are eye movements. Also, as mentioned in Section 4.2.1.6.1, eye-tracking data would provide insight into how top-down knowledge, like looking for the target link

at its expected location in the alphabetical list of links on a 3rd-level page, would direct exploration behavior. Lastly, as mentioned in Sections 4.2.2.2 and 4.2.3.1, eye-tracking data could provide insight into how people would look at and relate a heading to its group, and improve CogTool-Explorer in this aspect.

As explained in Section 4.2.1, I have chosen to work with the existing experiments and available participant data because the sequence of UI layouts (two-column, multi-page, half-flatten and then multi-group layout) used in this dissertation to develop and test CogTool-Explorer allowed some inference on the serial evaluation visual search process that takes place in the exploration tasks. However, the lack of eye-tracking data means that CogTool-Explorer does not and cannot claim to model and predict human behavior at this level of detail.

Therefore for future work, if resources are available, eye-tracking data should be collected to compare against model behavior and improve the sophistication of eye movement in CogTool-Explorer. Since the order in which options are looked at directly affects the selections made during exploration, a more sophisticated theory of eye movement would improve CogTool-Explorer's predictions of higher level task performance measures like *%Success* and *MeanClicksToSuccess*.

5.3 Layout-specific Knowledge

Another limitation is the specific knowledge about the half-flatten and multi-group layouts, as described in Sections 4.2.2.2 and 4.2.3.1, that has to be implemented in CogTool-Explorer 1.2 to enable the user model to understand how these layouts "work" as modeled in the UI mockup and behave appropriately. For example, in the half-flatten layout, the user model needs to know that the top-level and 2nd-level pages are actually continuations of one another and thus should continue exploration within the group of 2nd-level links whenever it transits to a 2nd-level page. Another example, in both the half-flatten and multi-group layouts, the user model needs to know the relationship between a group and its heading so that the user model would not treat them as two separate options and mistakenly select a heading link after just going back from its group.

In CogTool-Explorer 1.2, the above specific knowledge is implemented as individual "add-ons" and can be switched on or off. In fact, if all of the specific knowledge is switched off, and we run CogTool-Explorer 1.2 on the multi-page layout, it is exactly the same as running CogTool-Explorer 1.1 on the multi-page layout, because the generic knowledge about groups and group-based hierarchical exploration in CogTool-Explorer 1.2 will simply not kick in when the layout does not have groups. However, the need for such layout-specific knowledge in CogTool-Explorer 1.2 means that the model is currently less generic than desired; the model will still run with all of the layout-specific knowledge switched off but its behavior may be wrong.

A related limitation of CogTool-Explorer 1.2 is that the model currently lacks the knowledge to behave appropriately in commonly used layout schemas, such as Web links presented in alphabetical order or navigation links presented separately from the main content. It is reasonable to expect people would make use of these layouts, when present in the user-interface, to be more efficient during exploration, for example, knowing to start at the bottom of the alphabetical list when looking for something that starts with a "w" as opposed to a "b", or knowing to visually search

the navigation links first. For the CogTool-Explorer model to behave similarly, such layout specific knowledge would have to be added to the model.

A proposed solution for future work lies concurrently in a few places: (1) add to the current UI mockup facilities in CogTool to express these special relationships, like two frames are continuations of each another, that some widgets in these two frames are actually the same widget, or some widgets and/or groups have additional roles or properties like alphabetical ordering (2) update the device model with data structures to represent these relationships, and (3) update the user model to recognize and make use of these relationships. Identifying the common denominators in the specific knowledge and implementing them in a generic way will be the research challenge.

5.4 Tasks, Parameters and Generality

Another limitation is that CogTool-Explorer is developed and tested on a set of tasks, UI layouts and participant data from the AutoCWW experiments. Although the tasks and UI layouts were based on an actual encyclopedia website[15] (Figure 15), they are clearly not representative of the much wider range of website and webpage designs on the Web, and are further removed from the variety and richness of the graphical user-interface in devices ranging from desktop computers to handheld devices to aircraft cockpits.

I have chosen to work with the tasks, UI layouts and participant data from the AutoCWW experiments because they are suitable for investigating the factors (infoscent, layout position and grouping) that this dissertation focuses on, without confounding factors like graphics or animation. Using the AutoCWW experiments also facilitates the comparison between CogTool-Explorer and AutoCWW. However, this means that within the scope of this dissertation, CogTool-Explorer has not been tested on other tasks and UI layouts and its exploration process may not work as well. CogTool-Explorer also lacks the procedural knowledge for interacting with more complex widgets like pull-down lists and groups of checkboxes, which are already available in CogTool and are common in graphical user-interfaces. Furthermore, although the current model parameters values for *noise*, *k*, etc, work well, a full exploration of the parameter space would require computing resources beyond the scope of this dissertation. So, CogTool-Explorer still has some way to go before we can use it to model any UI layout and rely on its predictions in general.

The good news is that CogTool-Explorer has been successfully used to model other tasks in a different UI layout (John et al., 2009), that of a Control Display Unit in a Boeing 777's cockpit. For future work, CogTool-Explorer should be tested on other tasks and UI layouts and compared to new participant data. Advanced computational methods like Adaptive Mesh Refinement (Best et al., 2009) could also be used to do a full systematic exploration of the model parameter space.

5.5 Information Scent Scores

Another limitation is that CogTool-Explorer 1.2 has only been tested using infoscent scores derived from LSA cosine values computed with reference to the college-level TASA corpus (from

[15] http://encarta.msn.com (discontinued on October 31, 2009)

Touchstone Applied Science Associates, Inc.). While this is adequate and appropriate for the tasks, UI layouts and participant data used this dissertation, as AutoCWW also used the same, the TASA corpus may no longer be suitable for more contemporary tasks.

Imagine using CogTool-Explorer to model a user looking to access online videos on a smart-phone. We would expect an average computer user in the year 2011 to see a link or button labeled "YouTube" as attractive. However, "YouTube" does not even appear in the TASA corpus and would result in a null LSA cosine value. For CogTool-Explorer to become a practical tool for UI design and evaluation, it's sources of information scent scores have to keep in step with the real world. Future work could look into providing access to LSA semantic spaces based on other corpuses, or compute infoscent scores using PMI values based on search queries to large online document repositories like Wikipedia or even the Web.

5.6 Other Factors

There are factors listed in Table 1, namely goal formation, learning from exploration and knowledge about UI actions, and other factors such as cost-benefits of competing options and switching between multiple goals and sub-goals, that affect user exploration but are outside the scope of this dissertation. This does not mean that these factors are less important, but can be topics for future work.

Ultimately, we would want a model to predict the range of human behavior that would be observed in the real world when using an interactive system, on metrics such as number of errors and where they occur, performance time, learning time and what was learned, effects of fatigue, environmental factors, or emotion on performance, and even levels of satisfaction or joy when using the system. No computational model is up to that task at this writing.

By implementing CogTool-Explorer as a mechanistic process model in a widely established cognitive architecture like ACT-R (the same approach shared by Brumby and Howes' model (2004), SNIF-ACT 1.0, SNIF-ACT 2.0 and DOI-ACT), CogTool-Explorer has the longer-term potential of providing a research platform to add other factors that influence user exploration in an integrated way constrained by a cognitive architecture. As these factors interact in a mechanistic process model, interesting and potentially counterintuitive predictions have an opportunity to emerge.

REFERENCES

Amant, R. S., Riedel, M. O., Ritter, F. E., and Reifers, A. (2005). Image processing in cognitive models with SegMan. *Proceedings of HCI International '05, Volume 4 - Theories Models and Processes in HCI.* Paper #1869.

Anderson, J. R., and Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: Erlbaum.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004) An integrated theory of the mind. *Psychological Review 111*, 4, 1036-1060.

Best, B.J., Furjanic, C., Gerhart, N., Fincham, J. M., Gluck, K. A., Gunzelmann, G., and Krusmark, M., (2009). Adaptive Mesh Refinement for Efficient Exploration of Cognitive Architectures and Cognitive Models. *Proceedings of the 9th International Conference of Cognitive Modeling* (paper 252), Manchester, United Kingdom.

Blackmon, M. H., Polson, P. G., Kitajima, M., and Lewis, C. (2002). Cognitive walkthrough for the web. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '02). ACM, New York, NY, 463-470.

Blackmon, M. H., Kitajima, M., and Polson, P. G. (2003). Repairing usability problems identified by the cognitive walkthrough for the web. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, 497-504.

Blackmon, M. H., Kitajima, M., and Polson, P. G. (2005). Tool for accurately predicting website navigation problems, non-problems, problem severity, and effectiveness of repairs. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '05). ACM, New York, NY, 31-40.

Blackmon, M. H. (2011) AutoCWW [Website]. Accessed on January 17, 2009 at http://autocww.colorado.edu/~brownr/ACWW.php

Brumby, D. P., and Howes, A. (2003). Interdependence and past experience in menu choice assessment. *Proceedings of the 25th Annual Conference of the Cognitive Science Society,* Boston, MA, USA.

Brumby, D. P., and Howes, A. (2004). Good enough but I'll just check: Web page search as attentional refocusing. *Proceedings of the 6th Internal Conference on Cognitive Modeling.* Mahwah, NJ: Lawrence Erlbaum Associates.

Budiu, R., and Pirolli, P. L. (2007) Modeling navigation in degree of interest trees. *Proceedings of the 29th Annual Conference of the Cognitive Science Society,* Nashville, TN, USA.

Byrne, M. D., John, B. E., Wehrle, N. S., and Crow, D. C. (1999). The tangled Web we wove: a taskonomy of WWW use. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '99). ACM, New York, NY, 544-551.

Card, S. K., Moran, T.P., and Newell, A. (1980) The keystroke level model for user performance time with interactive systems. *Communications of the ACM 23, 7,* 396-410.

Card, S. K., Moran, T.P., and Newell, A. (1983) *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum Associates, Hillsdale, NJ, USA.

Chi, E. H., Pirolli, P., Chen, K., and Pitkow, J. (2001). Using information scent to model user information needs and actions and the Web. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '01). ACM, New York, NY, 490-497.

Chi, E. H., Rosien, A., Supattanasiri, G., Williams, A., Royer, C., Chow, C., Robles, E., Dalal, B., Chen, J., and Cousins, S. (2003). The bloodhound project: automating discovery of web usability issues using the InfoScentπ simulator. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, 505-512.

Chi, E. H. (2009) Bloodhound [Website]. Retrieved on August 14, 2009 from http://www2.parc.com/istl/groups/uir/projects/bloodhound/bloodhound.htm

Cox, A. L. (2002) *Exploratory Learning of Interactive Devices: What People Do and What People Learn*. Unpublished PhD, University of Hertfordshire, UK.

Cox, A. L., and Young, R. M. (2004). A rational model of the effect of information scent on the exploration of menus. *Poster session at the meeting of the 6th Internal Conference on Cognitive Modeling,* Pittsburgh, PA.

Fleetwood, M. D., and Byrne, M. D. (2006). Modeling the visual search of displays: A revised ACT-R/PM model of icon search based on eye tracking data. *Human-Computer Interaction, 21*(2), 153-197.

Franzke, M. (1994). *Exploration, acquisition, and retention of skill with display-based systems.* Ph.D. Thesis, Department of Psychology, University of Colorado, Boulder.

Franzke, M. (1995). Turning research into practice: characteristics of display-based interaction. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '95). ACM Press/Addison-Wesley Publishing Co., New York, NY, 421-428.

Fu, W.-T., and Pirolli, P. (2007). SNIF-ACT: A cognitive model of user navigation on the World Wide Web. *Human-Computer Interaction, 22,* 355-412.

Greeno, J. G., and Simon, H. A. (1988). Problem solving and reasoning. In R. C. Atkinson, R. Herrnstein, G. Lindzey, and R. D. Luce (Eds.), *Steven's handbook of experimental psychology,* 598-672. New York: Wiley.

Halverson, T., and Hornof, A. J. (2006). Towards a flexible, reusable model for predicting eye movements during visual search of text. *Proceedings of the 28th Annual Conference of the Cognitive Science Society,* Vancouver, B.C., Canada

Halverson, T., and Hornof, A. J. (2007). A minimal model for predicting visual search in human-computer interaction. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '07). 431-434.

Halverson, T. E. (2008). *An "active vision" computational model of visual search for human-computer interaction.* Doctoral Dissertation in Computer and Information Science. The University of Oregon, Eugene, OR, USA.

Halverson, T. E., and Hornof, A. J. (2008). The effects of semantic grouping on visual search. *CHI '08 Extended Abstracts on Human Factors in Computing Systems* (CHI '08). ACM, New York, NY, 3471-3476.

Hornof, A. J., and Halverson, T. (2003). Cognitive strategies and eye movements for searching hierarchical computer displays. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '03). ACM, New York, NY, 249-256.

Hornof, A. J. (2004). Cognitive strategies for the visual search of hierarchical computer displays. *Human-Computer Interaction, 19(3)*, 183-223.

John, B. E., and Kieras, D. E. (1996). Using GOMS for user interface design and evaluation: which technique? *ACM Transactions on Computer-Human Interaction, 3(4),* 287-319.

John, B. E., Prevas, K., Salvucci, D. D., and Koedinger, K. (2004). Predictive human performance modeling made easy. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '04). ACM, New York, NY, 455-462.

John, B. E., Blackmon, M. H., Polson, P. G., Fennell, K., and Teo, L. (2009) Rapid Theory Prototyping: An Example of an Aviation Task. *Proceedings of the Human Factors and Ergonomics Society 53rd Annual Meeting* (San Antonio, Texas, October 19-23, 2009).

John, B. E., (2010) Reducing the Variability between Novice Modelers: Results of a Tool for Human Performance Modeling Produced through Human-Centered Design. *Proceedings of the 19th Annual Conference on Behavior Representation in Modeling and Simulation (BRIMS)* (Charleston, SC, March 22-25, 2010).

John, B. E. (2011). CogTool (Version 1.1.3) [Software]. Available from http://cogtool.hcii.cs.cmu.edu/

Kieras, D. E., and Polson, P. G. (1985). An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22,* 365-394.

Kieras, D. E., Wood, S. D., and Meyer, D. E. (1997) Predictive engineering models based on the EPIC architecture for a multimodal high-performance human-computer interaction task. *ACM Transactions on Computer-Human Interaction, 4(3) (September 1997)*, 230-275.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review*, 95, 163-182.

Kintsch, W. (1998). *Comprehension: A paradigm for cognition.* Cambridge University Press.

Kitajima, M., and Polson, P. (1997). A comprehension-based model of exploration. *Human-Computer Interaction, 12, 4,* 345-389.

Kitajima, M., Soto, R., and Polson, P. G. (1998). LICAI+: A Comprehension-Based Model of The Recall of Action Sequences. In F. Ritter and R.M. Young (Eds.), *Proceedings of the Second European Conference on Cognitive Modelling* (pp. 82-89). Nottingham, UK: Nottingham University Press.

Kitajima, M., Blackmon, M.H., and Polson, P.G. (2000). A comprehension-based model of Web navigation and its application to Web usability analysis. In S. McDonald, Y. Waern and G. Cockton (Eds.), *People and Computers XIV - Usability or Else!* (Proceedings of HCI 2000, 357-373). Springer-Verlag.

Kitajima, M., Blackmon, M. H., and Polson, P. G. (2005). Cognitive Architecture for Website Design and Usability Evaluation: Comprehension and Information Scent in Performing by Exploration. *HCI International 2005.*

Kitajima, M., Polson, P. G., and Blackmon, M. H. (2007). CoLiDeS and SNIF-ACT: Complementary Models for Searching and Sensemaking on the Web. *Human Computer Interaction Consortium (HCIC) 2007 Winter Workshop.*

Landauer,T. K., McNamara, D. S., Dennis, S., and Kintsch W. (Eds). (2007). *Handbook of Latent Semantic Analysis,* Mahwah NJ: Lawrence Erlbaum Associates.

Lewis, C. H. (1986). A model of mental model construction. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '86). ACM, New York, NY, 306-313.

Lewis, C. H. (1988). How and why to learn why: Analysis-based generalization of procedures. *Cognitive Science, 12,* 211-256.

Miller, C. S., and Remington, R. W. (2004). Modeling Information Navigation: Implications for Information Architecture. *Human-Computer Interaction, 19,* 225-271.

Miller, G. A., Beckwith, R., Fellbaum, C. D., Gross, D., and Miller. K. (1990). WordNet: An online lexical database. *Int. J. Lexicograph. 3,* 4, (pp. 235-244). From http://en.wikipedia.org/wiki/WordNet

Misker, J., Taatgen, N. A., and Aasman, J. (2001). Validating a tool for simulating user interaction. *Proceedings of the Fourth International Conference on Cognitive Modeling.* Mahwah, NJ: Lawrence Erlbaum Associates, 163-168.

Morrison, J. B., Pirolli, P., and Card, S. K. (2001). A taxonomic analysis of what world wide web activities significantly impact people's decisions and actions. *CHI '01 Extended Abstracts on Human Factors in Computing Systems* (CHI '01). ACM, New York, NY, 163-164.

Pirolli, P. L., and Card, S. K. (1997). The evolutionary ecology of information foraging. PARC Technical Report. 1997 June. UIR-1997-01.

Pirolli, P., and Fu, W-T. (2003). SNIF-ACT: A model of information foraging on the world wide Web. In P. Brusilovsky, A. Corbett, and F. de Rosis (Eds.), *User Modeling 2003, 9th International conference on user modelling* (Vol. 2702, pp. 45–54). London: Springer-Verlag.

Pirolli, P. (2007). *Information Foraging Theory: Adaptive Interaction with Information.* Oxford University Press, USA.

Polson, P. G., and Lewis, C. (1990). Theory-based design for easily learned interfaces. *Human-Computer Interaction, 5(2-3),* 191-220.

Reeder, R. W., Pirolli P., and Card S. K. (2001) WebEyeMapper and WebLogger: tools for analyzing eye tracking data collected in web-use studies. *CHI '01 Extended Abstracts on Human Factors in Computing Systems* (CHI '01). ACM, New York, NY, USA, 19-20.

Rieman, J. (1994). *Learning strategies and exploratory behavior of interactive computer users.* Ph.D. Thesis, Technical Report CU-CS-723-94, Department of Computer Science, University of Colorado, CO, U.S.A.

Rieman, J. (1996). A field study of exploratory learning strategies. *ACM Transactions on Computer-Human Interaction, 3, 3,* 189-218.

Rieman, J., Young, R. M., and Howes, A. (1996). A dual-space model of iteratively deepening exploratory learning. *International Journal of Human-Computer Studies, 44, 6 (Jun. 1996),* 743-775.

Rosenholtz, R., Twarog, N. R., Schinkel-Bielefeld, N., and Wattenberg, M. (2009). An intuitive model of perceptual grouping for HCI design. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09). ACM, New York, NY, USA, 1331-1340.

Teo L., and John B. E. (2008). Towards a Tool for Predicting Goal-directed Exploratory Behavior. *Proceedings of the Human Factors and Ergonomics Society 52nd Annual Meeting*, 950-954.

Teo, L., and John, B. E. (2011), The Evolution of a Goal-Directed Exploration Model: Effects of Information Scent and GoBack Utility on Successful Exploration. *Topics in Cognitive Science*, 3: 154–165.

Wolfe, J. (1998). Visual Search. In H. Pashler (Ed.), *Attention,* 13-73. London, UK: University College London Press.

Young, R. M. (1998). Rational analysis of exploratory choice. In M.Oaksford and N.Chater (Eds.), *Rational models of cognition.* Oxford, UK: Oxford University Press.

Young, R. M., and Cox, A. L. (2000). A New Rational Framework for modeling Exploratory Device Learning ... but does it fit with ACT-R? *Proceedings of the 7th ACT-R Workship,* Carnegie Mellon University, Pittsburgh, PA, USA.

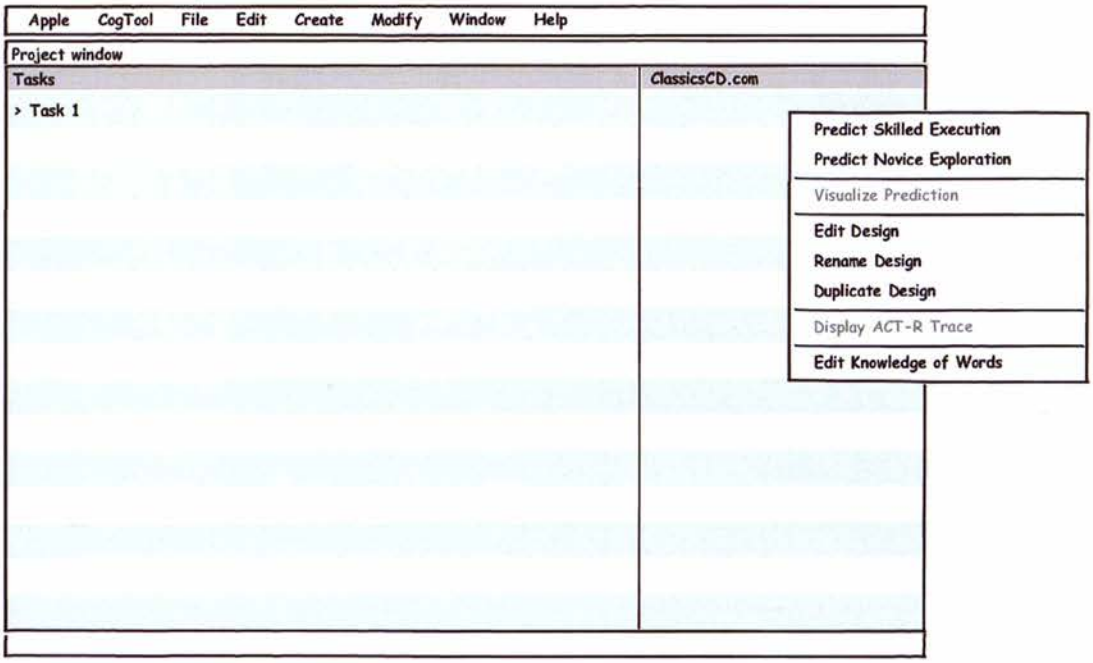
APPENDICES

Appendices A and B present the design outcomes from a series of design meetings that I conducted, between January to March 2010, with the CogTool software development team and Dr. Rachel Bellamy, a research collaborator from IBM's T. J. Watson Research Center. In these meetings, I proposed new designs and led the discussion, critique and redesign of various menus, dialogs and views that pertain to the integration of CogTool-Explorer into CogTool. While these design outcomes are not implemented into CogTool-Explorer in this dissertation, they represent a second design iteration toward making CogTool-Explorer into a practical tool for UI design and evaluation.

Appendix C contains the source code of the CogTool-Explorer 1.2 model.

Appendix D contains the source code of the Python script used to process the log file generated by the CogTool-Explorer 1.2 model.

A-1: CogTool Project Window



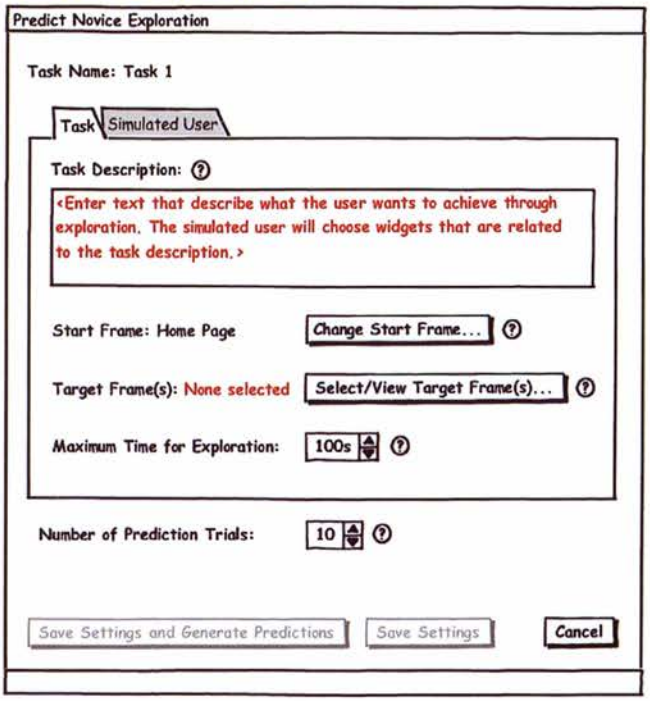
Existing CogTool UI

The figure above shows CogTool's Project Window. On start up, CogTool automatically creates a new CogTool project and a default "Task 1" in the Project Window. In this usage scenario, the modeler creates a "ClassicCD.com" website mockup. The modeler will rename "Task 1" with a more descriptive name in A-10.

UI Design for CogTool-Explorer

The modeler wants to model and get predictions of how a novice user might explore this website to accomplish a task. To do this, the modeler selects the "Predict Novice Exploration" menu item from the cell at the intersection of "Task 1" and "ClassicCD.com", as shown in the figure above.

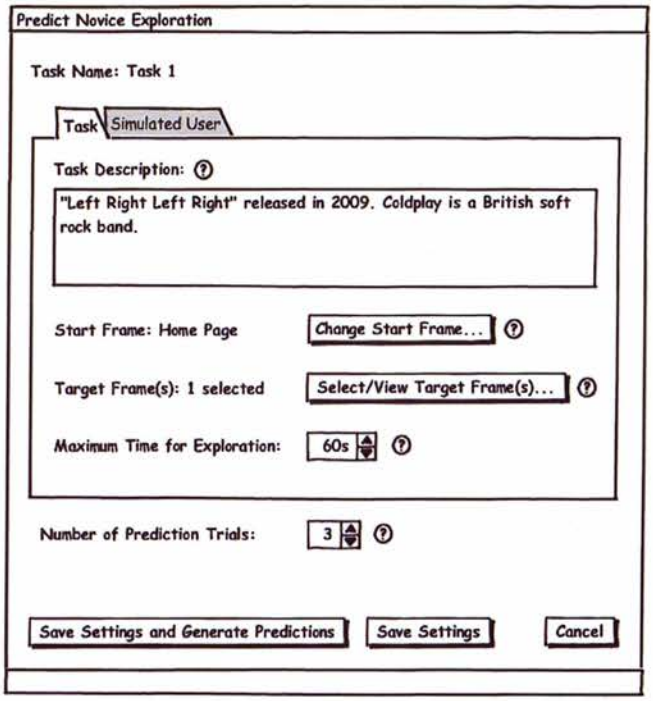
A-2: Predict Novice Exploration Dialog



UI Design for CogTool-Explorer

On selecting the "Predict Novice Exploration" menu item, the above dialog box appears for the modeler to set up CogTool-Explorer model runs. If this is the first time the dialog box appears for a particular task-mockup cell, or if the dialog box had appeared previously but no settings were changed and saved, the dialog box will have the default settings as shown in the figure above. The "Save Settings and Generate Predictions" and "Save Settings" buttons will become enabled after both the "Task Description" and "Target Frame(s)" fields are edited by the modeler.

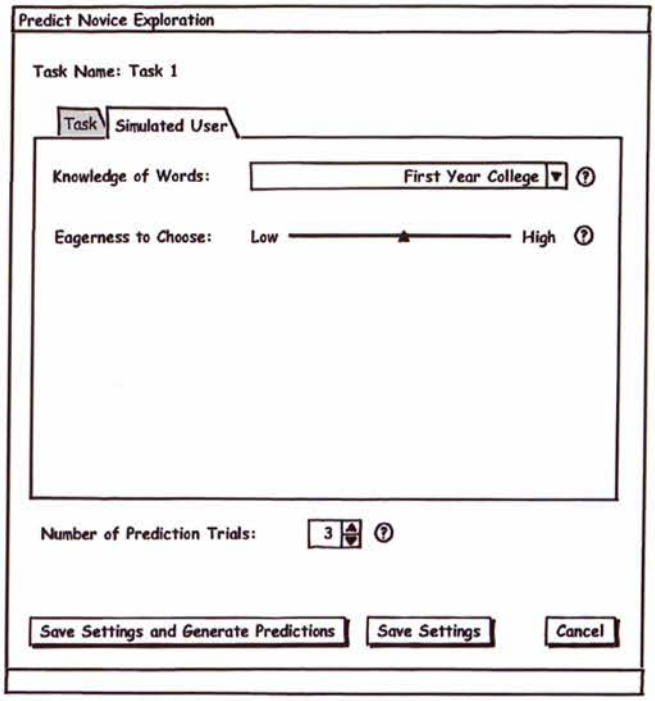
A-3: Predict Novice Exploration Dialog: Task Tab



UI Design for CogTool-Explorer

In this usage scenario, the modeler edits the "Task Description" field and specifies one frame in the UI mockup as a Target Frame. The modeler changes the "Maximum Time for Exploration" to 60 seconds and the "Number of Prediction Trials" to generate to 3, as shown in the figure above.

A-4: Predict Novice Exploration Dialog: Simulated User Tab



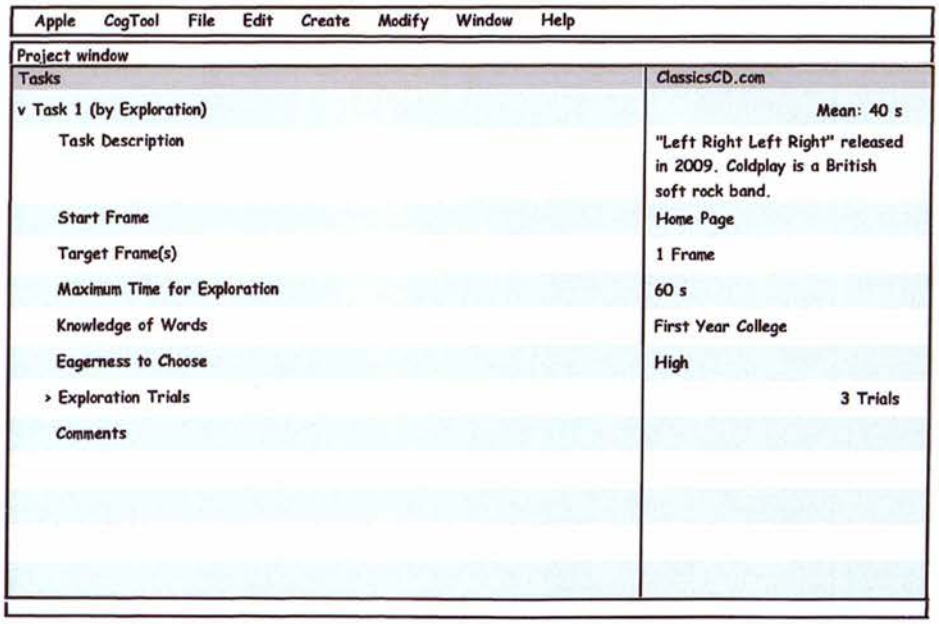
UI Design for CogTool-Explorer

The modeler next switches to the "Simulated User" tab as shown in the figure above. In the "Knowledge of Words" field, the modeler can select the appropriate LSA semantic space (corpus) to best match the demographics of the type of users the modeler wants predictions for. In the "Eagerness to Choose" field, the modeler can change parameter k in CogTool-Explorer (Section 4.1.5).

The "Save Settings and Generate Predictions" button will save the settings in this dialog box for this particular task-mockup cell and run CogTool-Explorer using these settings. The "Save Settings" button will save the settings in this dialog box for this particular task-mockup cell but will not run CogTool-Explorer. In both cases, the next time the "Predict Novice Exploration" dialog box appears for this particular task-mockup cell, the last saved settings will populate the fields in the dialog box.

In this usage scenario, the modeler increases the "Eagerness to Choose" to High and clicks on the "Save Settings and Generate Predictions" button.

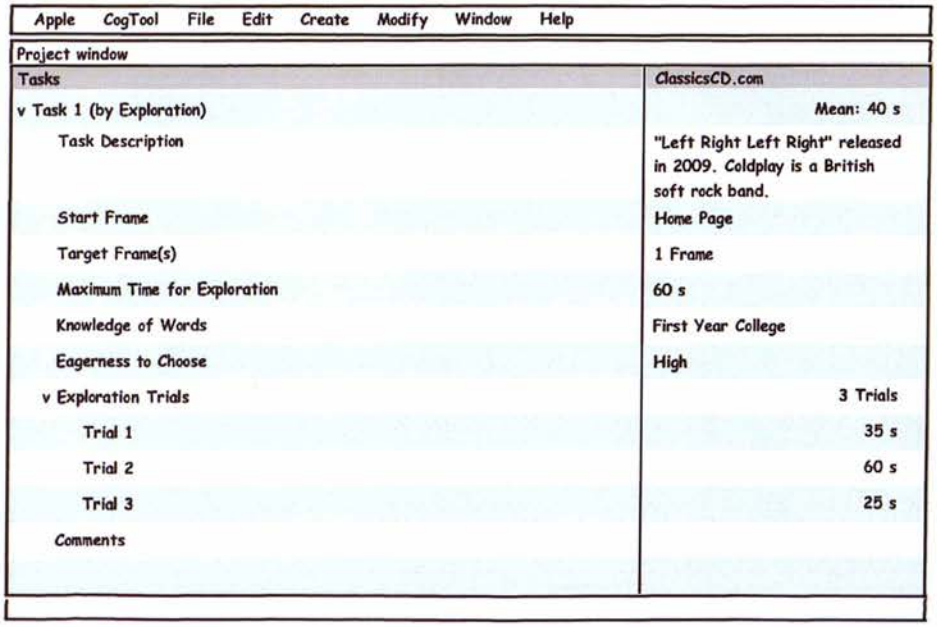
A-5: CogTool-Explorer Results and Settings



UI Design for CogTool-Explorer

The figure above shows what the Project Window looks like after the modeler clicked on the "Save Settings and Generate Predictions" button. The text "(by exploration)" is automatically appended to the original "Task 1" name to indicate that the results under this task came from running CogTool-Explorer (note that the modeler is free to rename the task, which the modeler does in A-10). The task is automatically expanded and the first six rows show the settings used to generate this set of CogTool-Explorer model runs. This is followed by the collection of the individual model runs, initially collapsed. The last row is an editable "Comments" field that the modeler can use to enter notes about the task.

A-6: CogTool-Explorer Model Runs

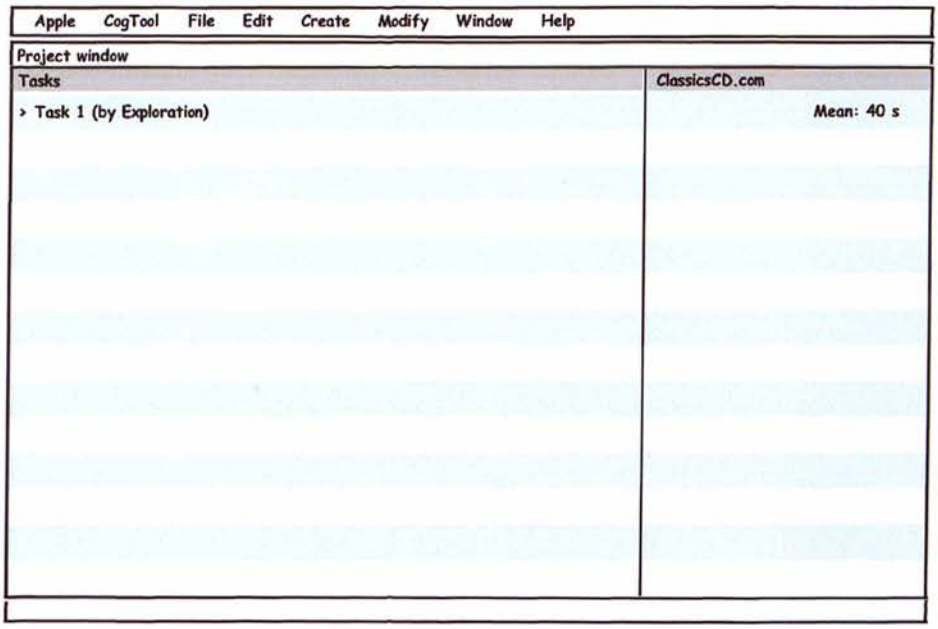


UI Design for CogTool-Explorer

To see the individual model runs, the modeler clicks on the expand/collapse icon next to "Exploration Trials". The figure above shows the fully expanded view of a set of CogTool-Explorer model runs in the Project Window.

A design goal for CogTool-Explorer is to reuse how CogTool presents KLM models using tasks and task groups in the Project Window, and extend the presentation style to show a set of CogTool-Explorer model runs and the settings used to generate them. For example, the collection of individual CogTool-Explorer model runs in "Exploration Trials" look just like a CogTool task group, but unlike a CogTool task group (A-8 to A-11 use a CogTool task group), its individual model runs are not editable, no model runs can be removed from the collection, and no model runs can be added to the collection unless they use the exact same settings. These restrictions are necessary to keep the collection of model runs consistent with its settings.

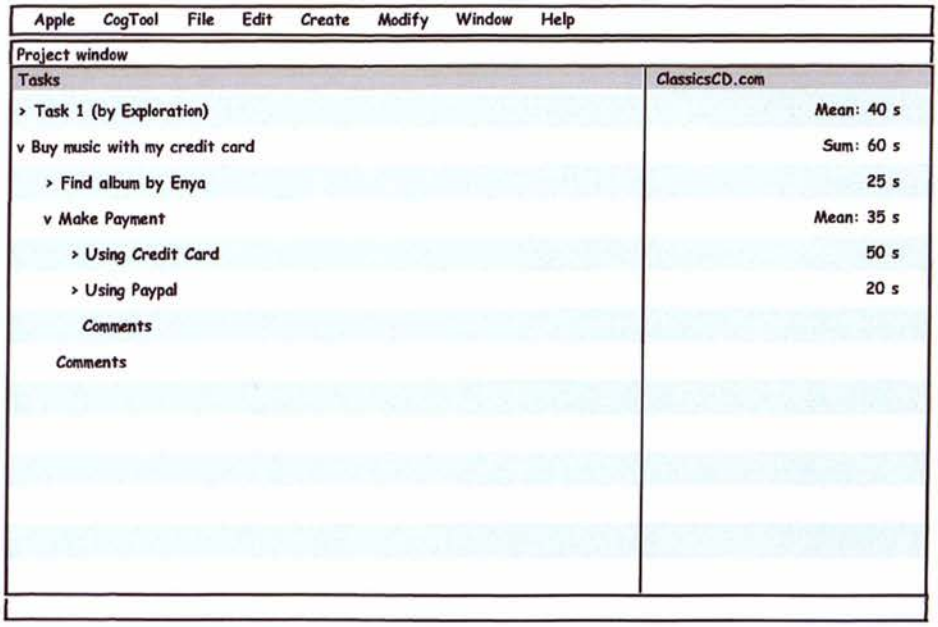
A-7: CogTool-Explorer Collapsed View



UI Design for CogTool-Explorer

The modeler clicks on the expand/collapse icon next to "Task 1 (by Exploration)" to collapse the task to take up just one row in the Project Window, as shown in the figure above.

A-8: CogTool Task Group

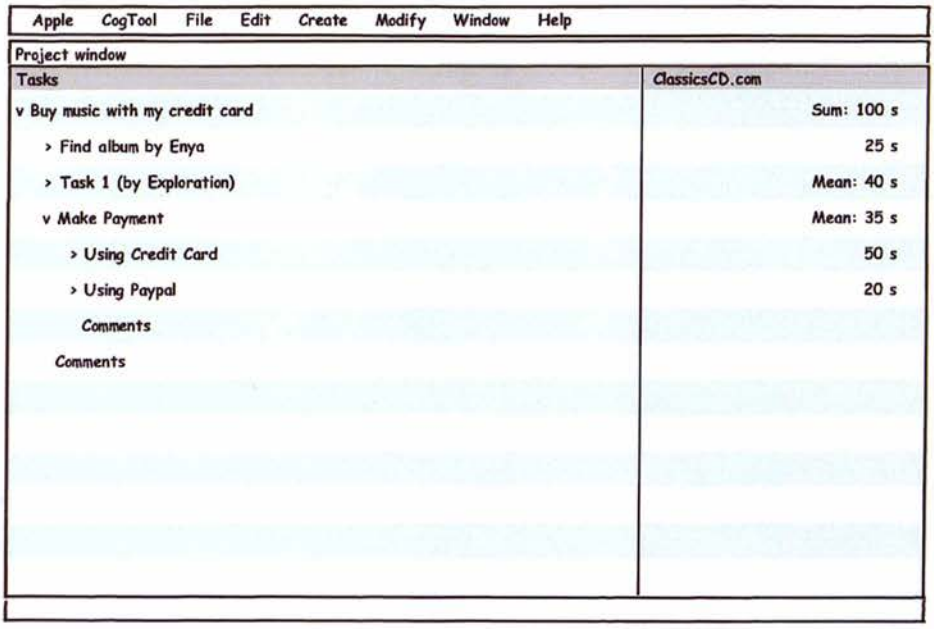


Existing CogTool UI

Using existing features in CogTool, the modeler starts to build up a more complex task in the "Buy music with my credit card" task group, as shown in the figure above. Inside this task group, the modeler creates a "Find album by Enya" KLM model and another task group named "Make Payment". Inside the "Make Payment" task group, the modeler creates two KLM models, "Using Credit Card" and "Using Paypal".

The "Buy music with my credit card" task group models a task with two sub-tasks: find a music album by Enya and then make payment. The "Make Payment" task group is set to show the mean task duration of its two KLM models, "Using Credit Card" and "Using Paypal", thus modeling that users of ClassicCD.com would pay with a credit card half the time and the other half of the time pay with Paypal. The "Buy music with my credit card" task group is set to show the sum of the durations of the "Find album by Enya" task and the "Make Payment" task group, thus modeling that these two sub-tasks happen sequentially.

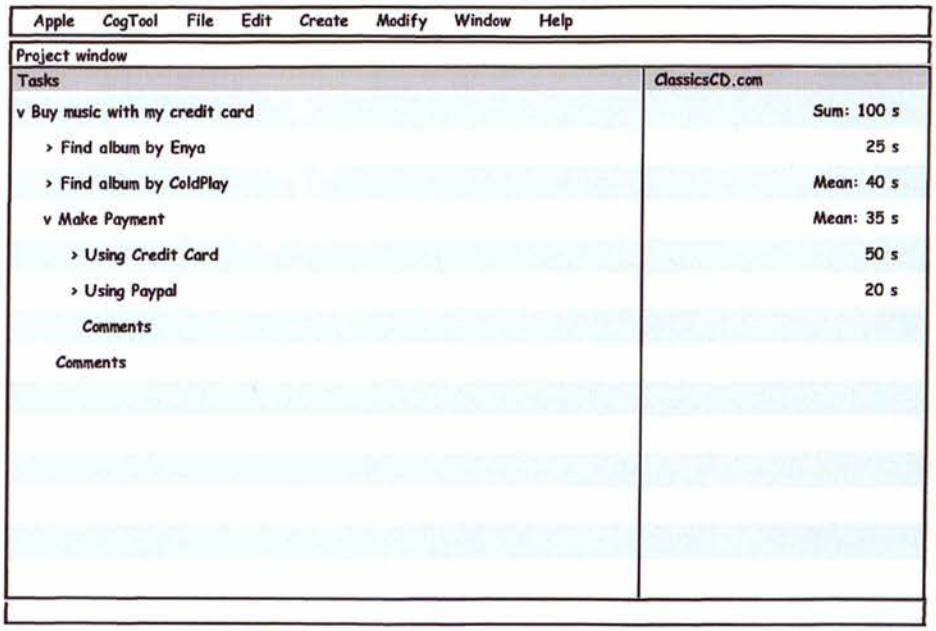
A-9: CogTool Task Group with CogTool-Explorer Task



Existing CogTool UI

Using existing interaction techniques in CogTool, the modeler drags and drops "Task 1 (by Exploration)" into the "Buy music with my credit card" task group, as shown in the figure above. Note that the "Buy music with my credit card" task group updates its sum of task durations accordingly.

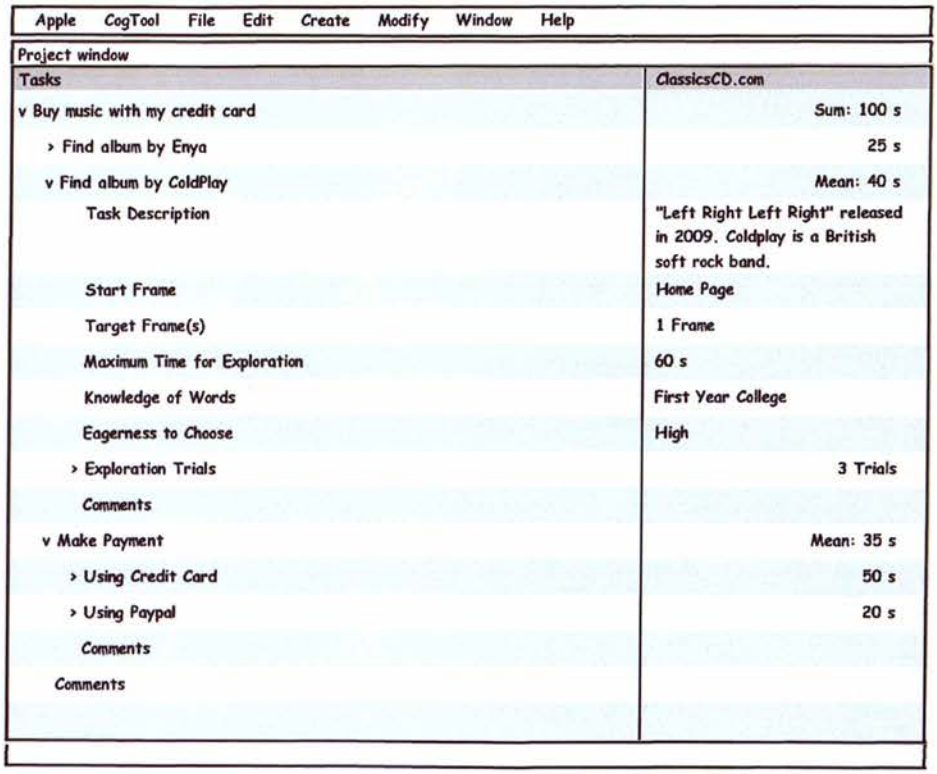
A-10: RENAME COGTOOL-EXPLORER TASK



Existing CogTool UI

The modeler renames "Task 1 (by Exploration)" to "Find album by ColdPlay" as shown in the figure above, thus, builds up the "Buy music with my credit card" task group to model a user of ClassicCD.com finding two music albums and then making payment.

A-11: CogTool-Explorer Expanded View

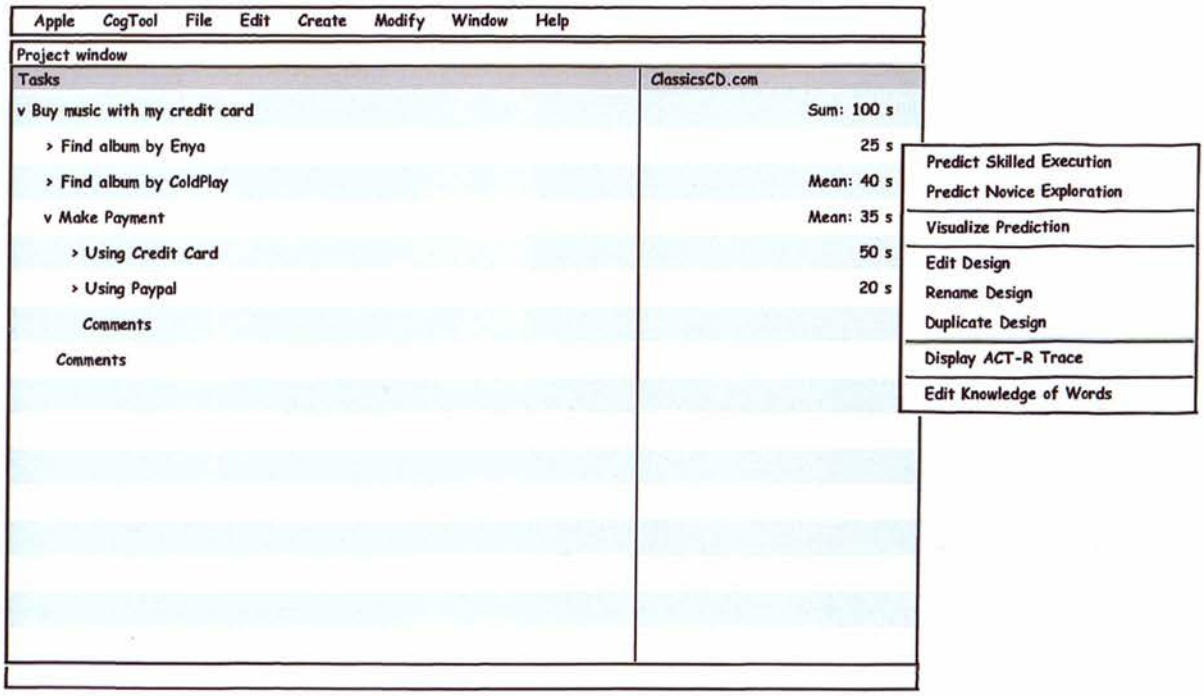


UI Design for CogTool-Explorer

The modeler clicks on the expand/collapse icon next to "Find album by ColdPlay" to look at the settings again, as shown in the figure above

The "Buy music with my credit card" task group now models a task with three sub-tasks: find a music album by Enya, find a music album by ColdPlay, and then make payment. "Find album by Enya" models a user of ClassicCD.com following a particular path through the UI mockup, that is, the user knows exactly what to do. "Find album by ColdPlay" models a user who does not know exactly what to do and has to explore the website, thus, it shows the mean duration of its collection of CogTool-Explorer model runs. As explained in A-8, "Make Payment" models users of ClassicCD.com would pay using a credit card half the time and the other half of the time pay using Paypal.

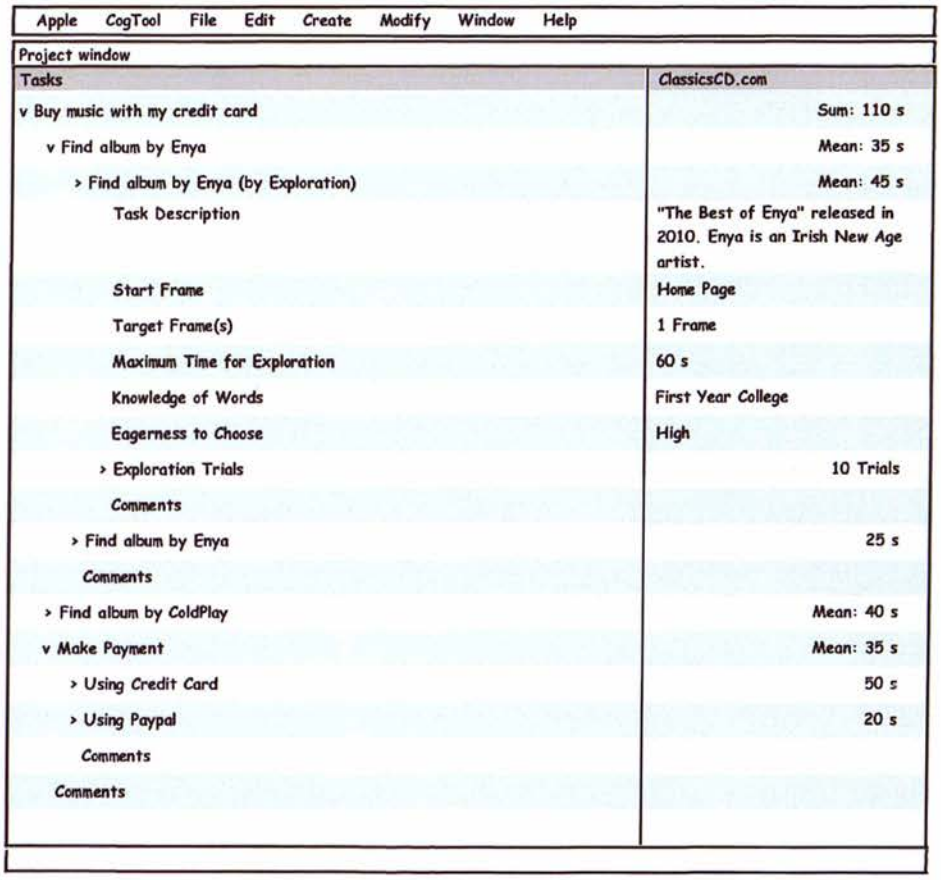
A-12: Predict Novice Exploration from Existing KLM Model



UI Design for CogTool-Explorer

The modeler wonders what if some users do not know where the Enya album is in ClassicCD.com? To proceed, the modeler selects the "Predict Novice Exploration" menu item from the cell at the intersection of "Find album by Enya" and "ClassicCD.com", as shown in the figure above. This brings up the "Predict Novice Exploration" dialog box, shown in A-2 to A-4, for the modeler to set up CogTool-Explorer model runs.

A-13: Automated Creation of Task Group



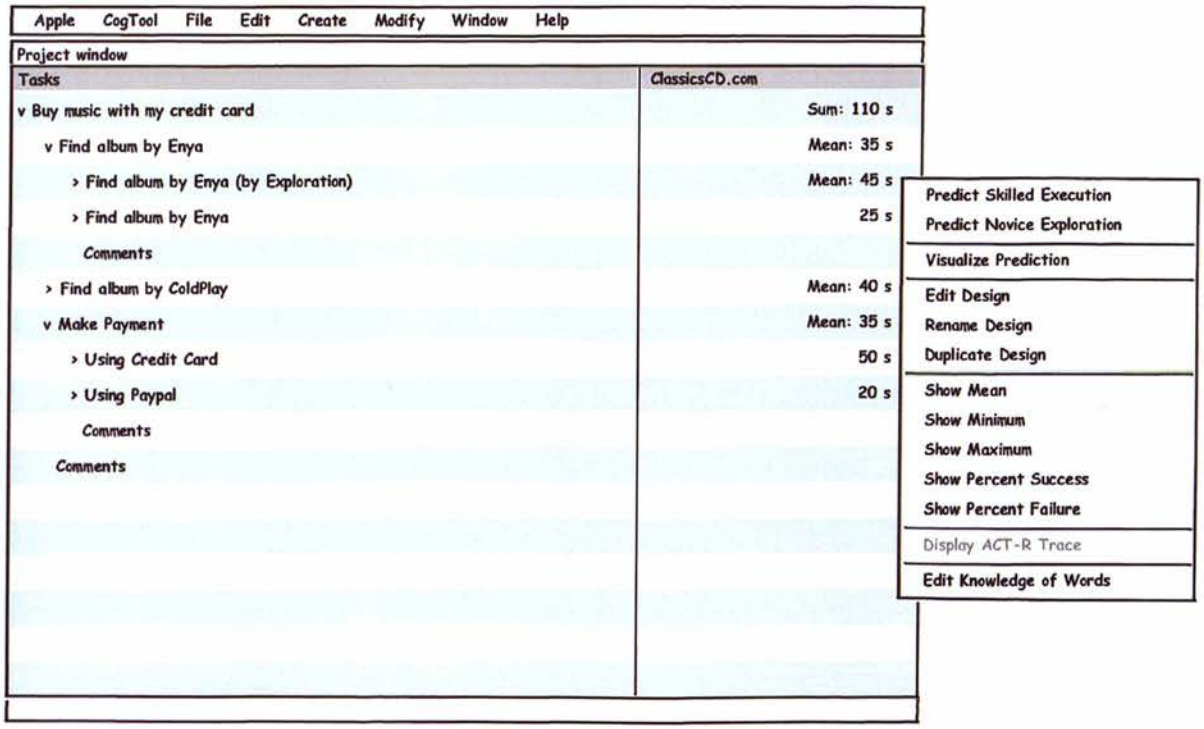
UI Design for CogTool-Explorer

The figure above shows what the Project Window looks like after the modeler clicked on the "Save Settings and Generate Predictions" button in the "Predict Novice Exploration" dialog box. Like in A-5, the text "(by exploration)" is automatically appended to the original "Find album by Enya" name to indicate that the results shown under this task came from running CogTool-Explorer. The task is automatically expanded and the first six rows show the settings used to generate this set of CogTool-Explorer model runs. This is followed by the collection of the individual model runs, initially collapsed. The last row is an editable "Comments" field that the modeler can use to enter notes about the task.

Unlike in A-5 where "Predict Novice Exploration" was selected from an empty cell, here in A-12, "Predict Novice Exploration" was selected from a cell that holds the "25s" task duration result from the "Find album by Enya" KLM model, thus, a new "Find album by Enya" task group is automatically created and consists of the original "Find album by Enya" KLM model and the new "Find album by Enya (by Exploration)", as shown in the above figure.

The "Find album by Enya" task group is also automatically set to show the mean task duration, thus modeling that half of the users of ClassicCD.com know where the Enya album is in ClassicCD.com while the other half do not and need to explore ClassicCD.com.

A-14: Predict Novice Exploration from Existing CogTool-Explorer Task



UI Design for CogTool-Explorer

Between A-12 and A-13, the modeler selected "First Year College" in the "Knowledge of Words" field. Now, the modeler wonders what if some users of ClassicCD.com have a 12th grade vocabulary?

To proceed, the modeler selects the "Predict Novice Exploration" menu item from the cell at the intersection of "Find album by Enya (by Exploration)" and "ClassicCD.com", as shown in the figure above. This brings up the "Predict Novice Exploration" dialog box, shown in A-2 to A-4, for the modeler to set up CogTool-Explorer model runs. Note that because the modeler had previously changed and saved settings for this particular cell between A-12 and A-13, the saved settings will now pre-populate the fields in the "Predict Novice Exploration" dialog box. The modeler then switches to the "Simulated User" tab, shown in A-4, selects "12th Grade" in the "Knowledge of Words" field, and clicks on the "Save Settings and Generate Predictions" button.

A-15: CogTool-Explorer Results and Settings

Apple CogTool File Edit Create Modify Window Help

Project window

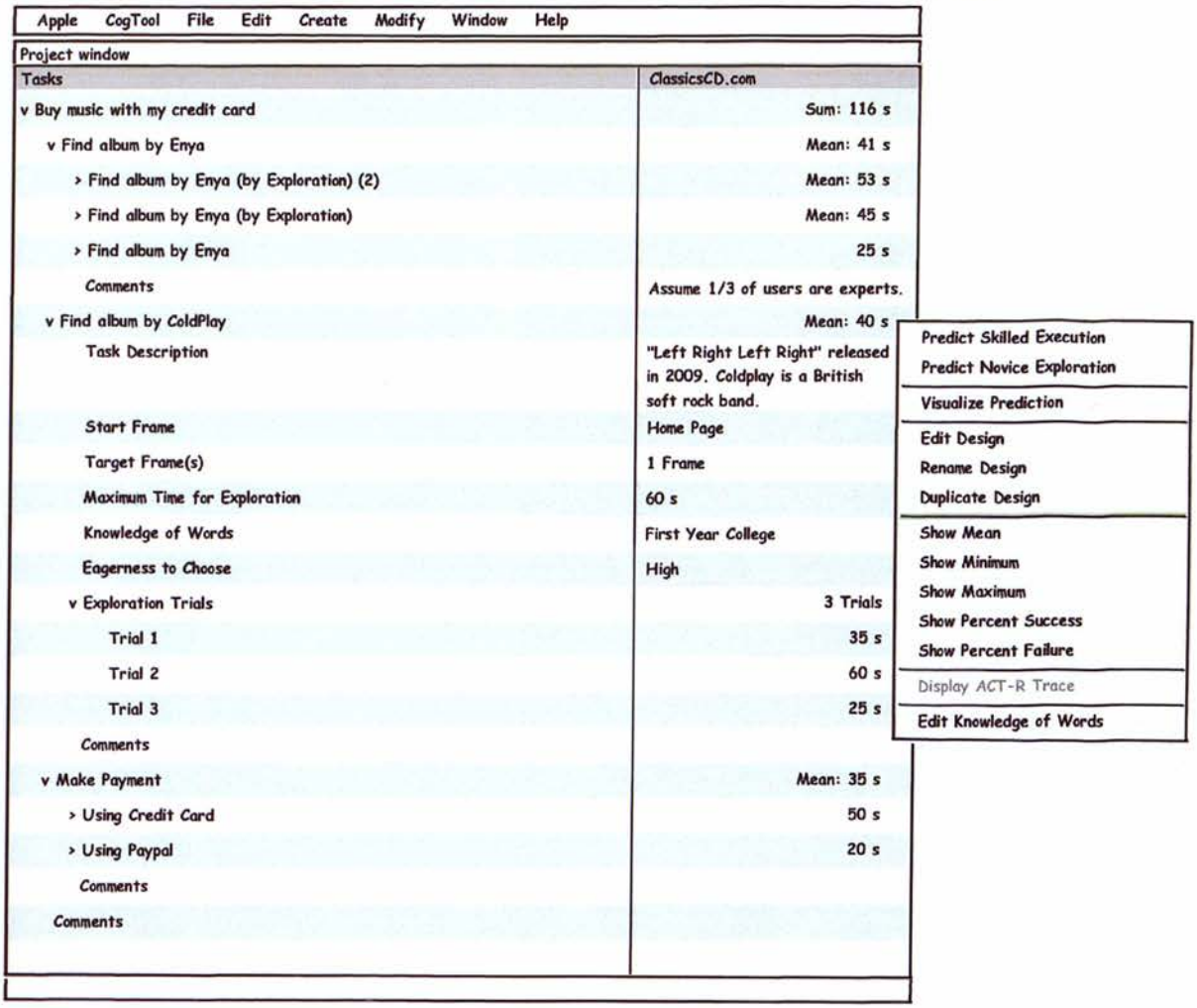
Tasks	ClassicsCD.com
v Buy music with my credit card	Sum: 116 s
v Find album by Enya	Mean: 41 s
> Find album by Enya (by Exploration) (2)	Mean: 53 s
Task Description	"The Best of Enya" released in 2010. Enya is an Irish New Age artist.
Start Frame	Home Page
Target Frame(s)	1 Frame
Maximum Time for Exploration	60 s
Knowledge of Words	12th Grade
Eagerness to Choose	High
> Exploration Trials	10 Trials
Comments	
> Find album by Enya (by Exploration)	Mean: 45 s
> Find album by Enya	25 s
Comments	Assume 1/3 of users are experts.
> Find album by ColdPlay	Mean: 40 s
v Make Payment	Mean: 35 s
> Using Credit Card	50 s
> Using Paypal	20 s
Comments	
Comments	

UI Design for CogTool-Explorer

The figure above shows what the Project Window looks like after the modeler clicked on the "Save Settings and Generate Predictions" button in the "Predict Novice Exploration" dialog box. Since the settings were changed, a new set of CogTool-Explorer model runs is created in the Project Window, and because the task name "Find album by Enya (by exploration)" already exists in the "Find album by Enya" task group, CogTool automatically appends the text "(2)" to the original "Find album by Enya (by exploration)" name for this new set of CogTool-Explorer model runs.

Now, the "Find album by Enya" task group models a third of the users of ClassicCD.com know where the Enya album is in ClassicCD.com while the other two-thirds do not and need to explore ClassicCD.com, with half of them having a first year college vocabulary and the other half having a 12th grade vocabulary. The modeler makes a note about this in the "Comments" area for the "Find album by Enya" task group, as shown in the figure above.

A-16: Change Result Summary Format for CogTool-Explorer Task



UI Design for CogTool-Explorer

The above figure shows what the Project Window looks like after the modeler collapses "Find album by Enya (by Exploration) (2)" and expands "Find album by ColdPlay".

The modeler wants to select a different result summary for "Find album by ColdPlay". To proceed, the modeler selects "Show Percent Success" from the cell "Mean: 40 s".

A-17: Percent Success of CogTool-Explorer Task

Apple CogTool File Edit Create Modify Window Help

Project window

Tasks	ClassicsCD.com
v Buy music with my credit card	Sum: 116 s
v Find album by Enya	Mean: 41 s
> Find album by Enya (by Exploration) (2)	Mean: 53 s
> Find album by Enya (by Exploration)	Mean: 45 s
> Find album by Enya	25 s
Comments	Assume 1/3 of users are experts.
v Find album by ColdPlay	Success: 67% of 3 Trials
Task Description	"Left Right Left Right" released in 2009. Coldplay is a British soft rock band.
Start Frame	Home Page
Target Frame(s)	1 Frame
Maximum Time for Exploration	60 s
Knowledge of Words	First Year College
Eagerness to Choose	High
v Exploration Trials	3 Trials
Trial 1	Success
Trial 2	Failure
Trial 3	Success
Comments	
v Make Payment	Mean: 35 s
> Using Credit Card	50 s
> Using Paypal	20 s
Comments	
Comments	

UI Design for CogTool-Explorer

The figure above shows what the Project Window looks like after the modeler select "Show Percent Success" for "Find album by ColdPlay". The result summary shows that two-thirds of the model runs were successful, that is, they reached the target frame before the maximum time for exploration is up, whereas the rest of the model runs failed to reach the target frame before the maximum time for exploration is up. The format for the individual model run result is also updated to either "Success" or "Failure" to match the format of the result summary.

A-18: CogTool-Explorer Expanded View and Comments Fields

Apple CogTool File Edit Create Modify Window Help

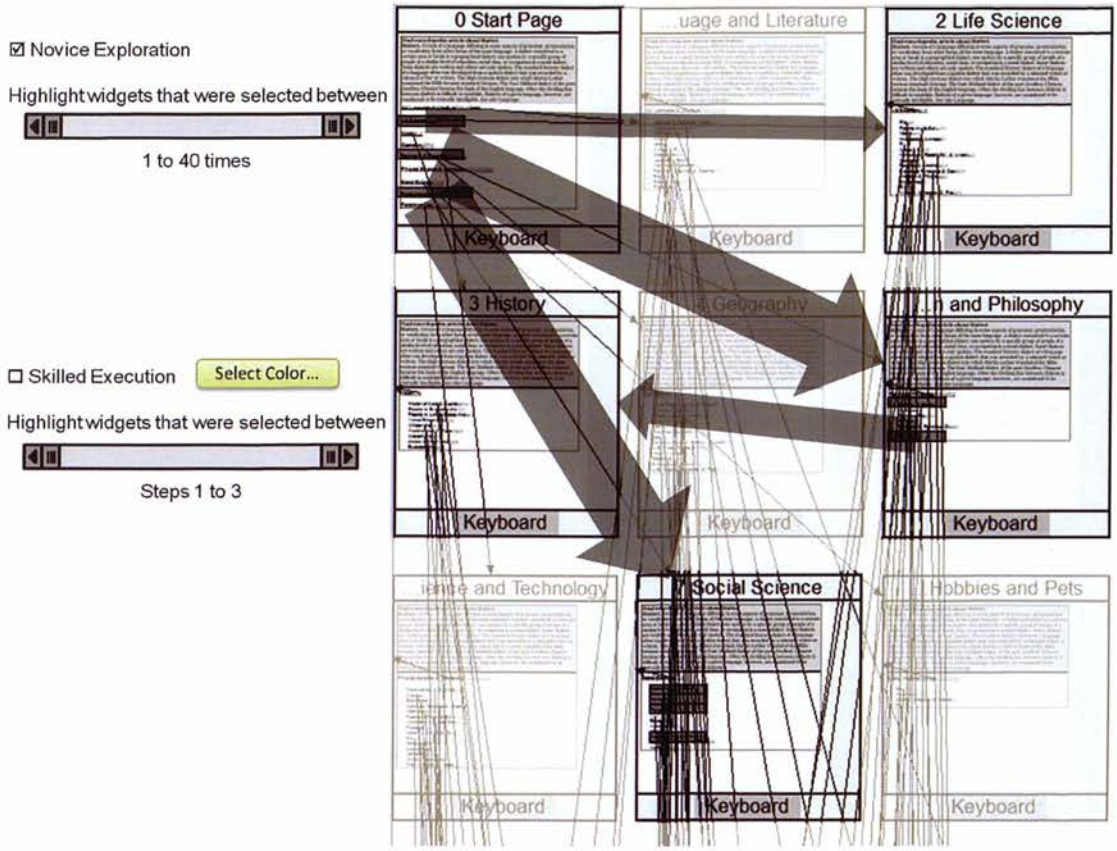
Project window

Tasks	ClassicsCD.com
v Buy music with my credit card	Sum: 116 s
v Find album by Enya	Mean: 41 s
> Find album by Enya (by Exploration) (2)	Mean: 53 s
> Find album by Enya (by Exploration)	Mean: 45 s
v Find album by Enya	25 s
Comments	Expert user know the site and where to find album by Enya.
Comments	Assume 1/3 of users are experts.
v Find album by ColdPlay	Success: 67% of 3 Trials
Task Description	"Left Right Left Right" released in 2009. Coldplay is a British soft rock band.
Start Frame	Home Page
Target Frame(s)	1 Frame
Maximum Time for Exploration	60 s
Knowledge of Words	First Year College
Eagerness to Choose	High
v Exploration Trials	3 Trials
Trial 1	Success
Trial 2	Failure
Trial 3	Success
Comments	
v Make Payment	Mean: 35 s
v Using Credit Card	50 s
Comments	Takes longer to enter Credit Card number.
v Using Paypal	20 s
Comments	Is faster to enter email address.
Comments	Assume half pay by Credit Card and half pay by Paypal.
Comments	

UI Design for CogTool-Explorer

The figure above shows what the Project Window looks like after the modeler expands the KLM models "Find album by Enya", "User Credit Card" and "Using Paypal", and enters notes in their "Comments" rows.

B-1: Visualization of CogTool-Explorer Model Runs



UI Design for CogTool-Explorer

B-1 to B-7 show a design for visualizing a set of CogTool-Explorer model runs. The modeler would get to this visualization by selecting "Visualize Prediction" from the results cell of a set of CogTool-Explorer model runs (see A-16 for an example).

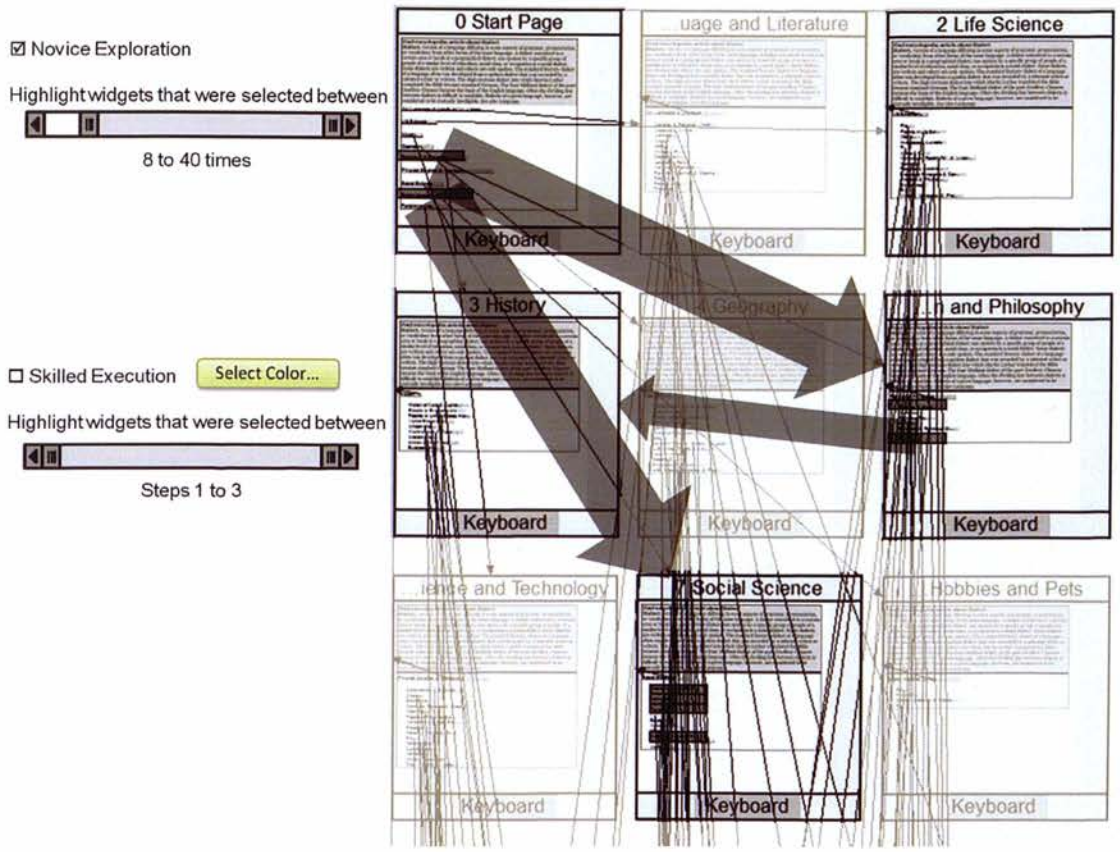
The visualization reuses CogTool's Design Window (Figure 41b) to show the frames and transitions in a UI mockup, but renders the UI mockup in grayscale[16] and fades out the frames that were never visited in this set of CogTool-Explorer model runs, as shown in the figure above. To show the widgets that were selected and the transitions that were followed in this set of CogTool-Explorer model runs, the visualization overlays gray rectangles over widgets that were selected and gray block arrows over transitions that were followed, with the relative widths of the block arrows showing the number of times each widget was selected.

The visualization defaults to highlighting all the widgets that were selected in this set of CogTool-Explorer model runs.

---

[16] The figures in B-1 to B-7 are created by editing screen captures of the existing CogTool's Design Window. The transitions represented by thin line arrows should be faded out in the figures, but due to constraints in image editing, segments of these thin line arrows are not faded when they overlap non-faded frames.

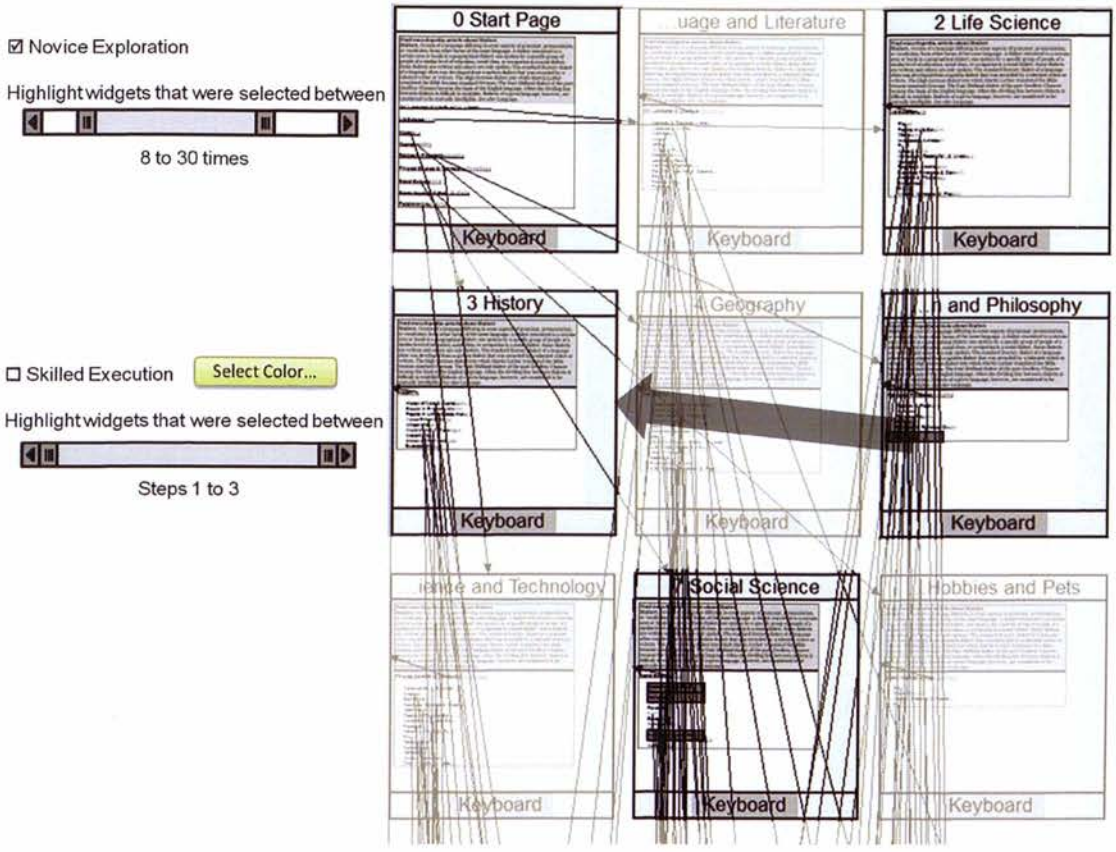
B-2: Visualization of CogTool-Explorer Model Runs: Filtered #1



UI Design for CogTool-Explorer

To filter the highlighting of widgets that were selected in this set of CogTool-Explorer model runs, the modeler can use the range slider as shown in the figure above. In this example, the modeler wants to focus on the widgets that were selected more often, and moves the left slider control to highlight only those widgets that were selected at least 8 times in this set of CogTool-Explorer model runs.

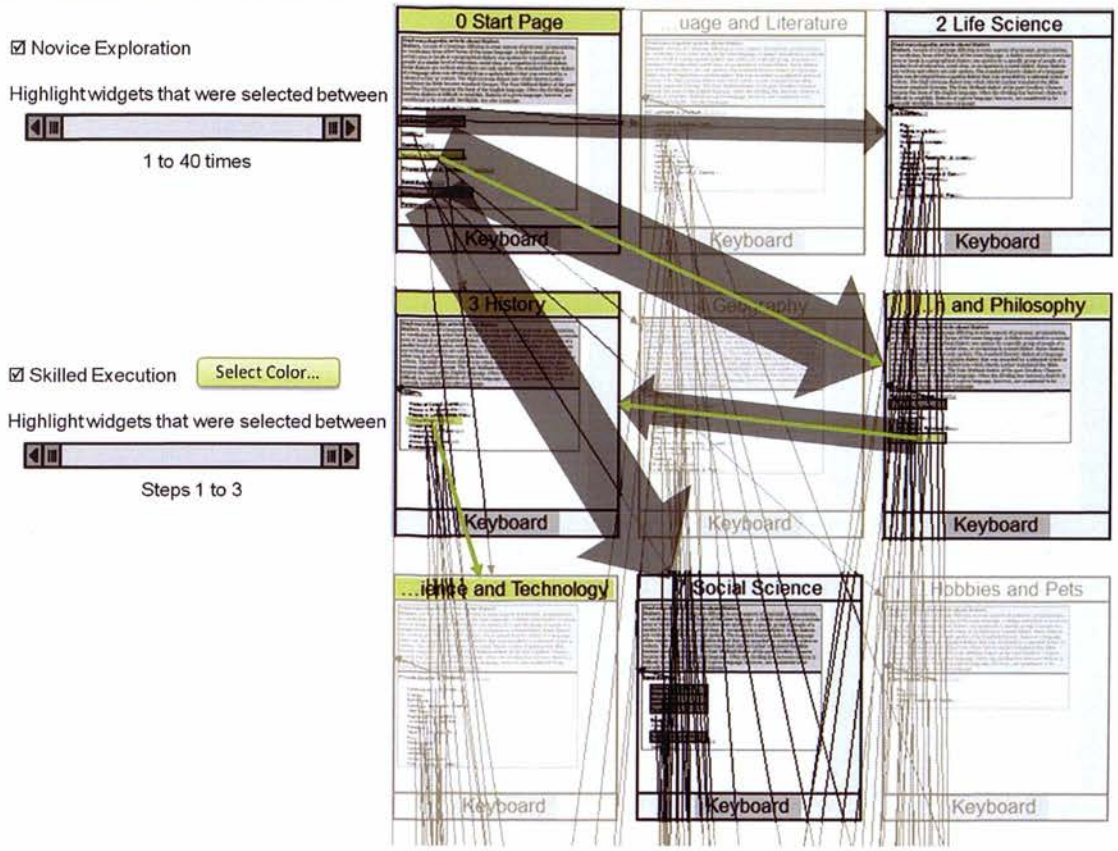
B-3: Visualization of CogTool-Explorer Model Runs: Filtered #2



UI Design for CogTool-Explorer

The range slider also allows the specification of an upper limit on the widgets to highlight. In this example, the modeler moves the right slider control to highlight only those widgets that were selected 30 times or less in this set of CogTool-Explorer model runs, as shown in the figure above.

B-4: COMPARISON OF COGTOOL-EXPLORER MODEL RUNS

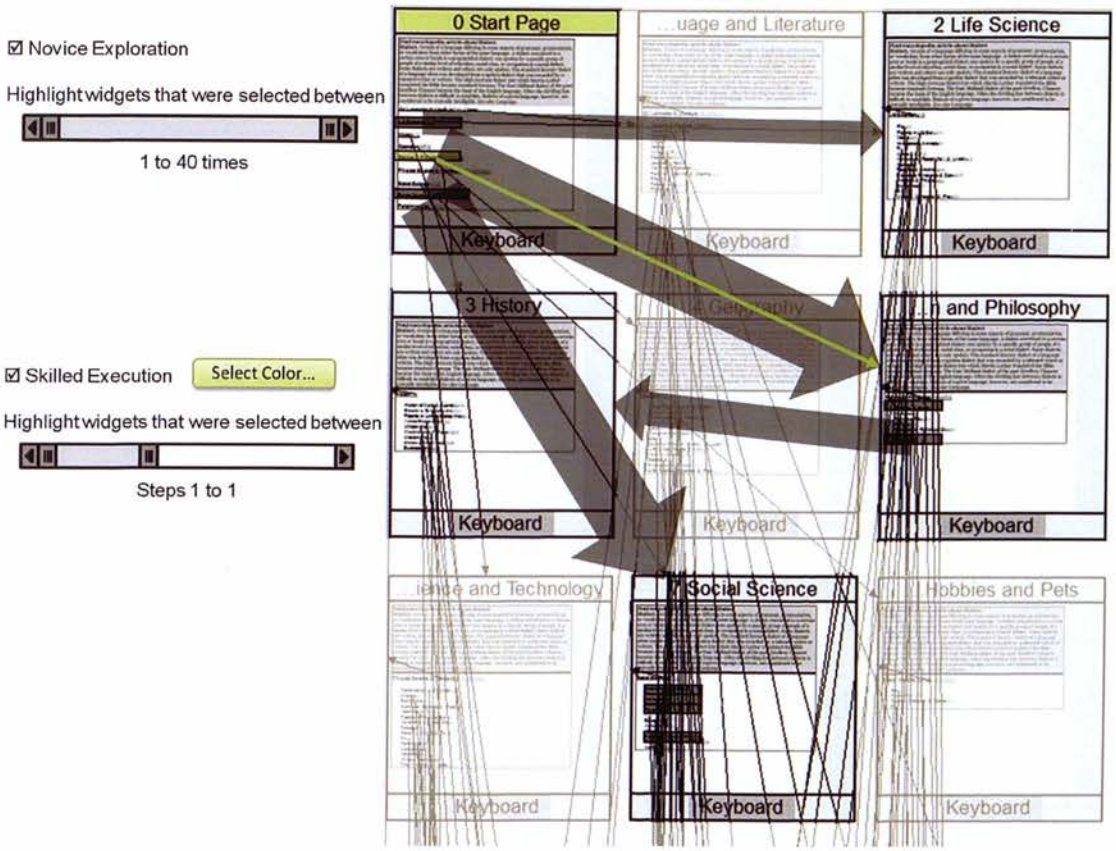


UI Design for CogTool-Explorer

To compare this set of CogTool-Explorer model runs to a particular model run or a KLM model from the same UI mockup, the modeler can check the "Skilled Execution" box and select a particular model run or a KLM model to show the path taken by that model in the visualization.

In this example, the modeler first adjusts the range sliders for "Novice Exploration" to highlight all widgets that were selected in this set of CogTool-Explorer model runs. The modeler next checks the "Skilled Execution" box and selects a KLM model from the same UI mockup (selection dialog not shown in the figure above). The visualization then highlights in color the path taken by the selected model as shown in the figure above.

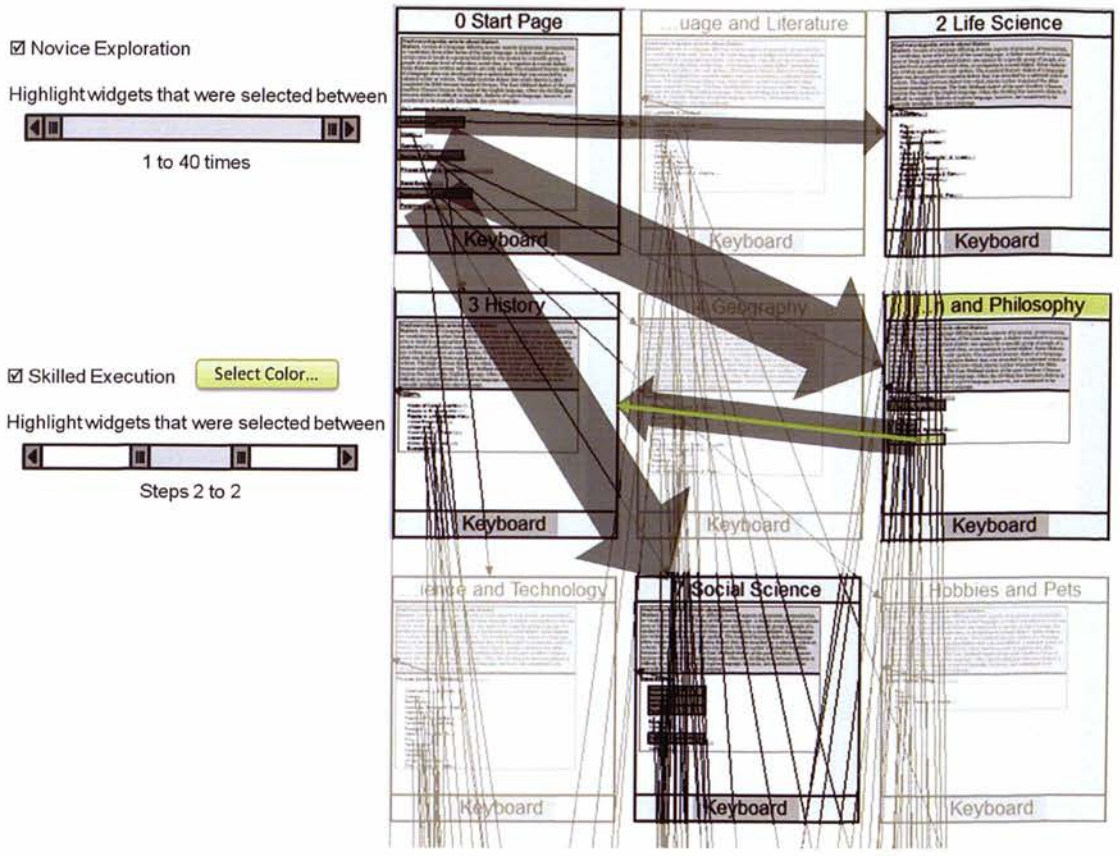
B-5: Comparison of CogTool-Explorer Model Runs: Filtered #1



UI Design for CogTool-Explorer

The range slider for "Skilled Execution" allows the modeler to selectively highlight only a segment of the path taken by the selected model. In this example, the modeler moves the right slider control to highlight only the widget that was selected in the first step of the path, as shown in the figure above.

B-6: Comparison of CogTool-Explorer Model Runs: Filtered #2



UI Design for CogTool-Explorer

In the figure above, the modeler moves the range slider bar to the center, to highlight only the widget that was selected in the second step of the path.

```
;;; Model refinements
(defparameter *modified-mean-prev-page* t)
(defparameter *modified-mean-current-page* t)
(defparameter *update_confidence* t)
(defparameter *use-back-button* t) ;;; t means model will look for a Back button and click on it to go-back, nil means model can
"magically" go back to the previous frame it came from.

(defparameter *mixed-visual-search* nil) ;;; t means model may choose to focus next on the nearest group instead of doing a strict
header-first process
(defparameter *non-header-process-conf-scale* 3) ;;; numeric value only applies if  *mixed-visual-search* is t

(defparameter *use-back-button-history-in-half-flatten-layout* t) ;;; in the half-flatten layout, back buttons have no transitions, so
have to use exploration history when set to t
(defparameter *recognize-equivalent-links-in-half-flatten-layout* t)
(defparameter *handle-groups-in-half-flatten-layout* t)
(setf *2nd-level-pages* '("Art, Language & Literature" "Life Science" "History" "Geography" "Religion & Philosophy" "Physical Science
& Technology" "Social Science" "Sports, Hobbies, & Pets" "Performing Arts"))

(defparameter *consider-familarity* nil)
(defparameter *modified-previously-chosen-widgets* nil)
(defparameter *infoscent_variabilty_between_runs* nil)
(defparameter *second-best-prev-page* nil)

(defparameter *corrected-k-update* t)

;;; Model parameters
(defparameter *similarity-scale-factor* 50)
(defparameter *infoscent-noise* 1)
(defparameter *go-back-cost* 1)

;;; Task parameters
(defparameter *CT-E_timeout* 130)

;;; Model log file
(defparameter *file* nil)

(defparameter *model-run* 0)

(defparameter *CT-E-Debug* nil)

(defun reset-trial-parameters () ; parameters to be reset at the start of each new trial or run of the model
       (setf *widget-infoscent-stack* nil) ; list of lists of infoscent values seen on previous pages or frames
       (setf *history* nil) ; enable go-back to previous page without back button on page
       (setf *current-group* nil) ; for ease of printing to model log file
       (setf *step-index* 1)
       (set-visloc-default isa cogtool-visual-location
              member-of-group       nil
              - display-label       "Back"
              :attended             new
```

```
            :nearest                current)
        (if *infoscent_variabilty_between_runs* (setf *assessed-infoscent* (make-hash-table :test #'equalp)))
)

(defun reset-frame-parameters () ; parameters to be reset at the start of each visit to a page or frame
        (spp read-another-widget :u 0)
        (spp choose-best-widget :u 0)
        (spp go-back :u 0)
        (setf *widget-infoscent-list* nil) ; list of infoscent values seen on this frame or group
        (setf *number-widgets-read* 0)
        (setf *best-widget-scent* 0)
        (setf *current-widget-scent* 0)
        (remove-visual-finsts)
)

(defparameter *overridden-global-parameters*
        '(
        :visual-finst-span 100 ;;; "Perfect" visual search memory for each frame
        :visual-num-finsts 100 ;;; "Perfect" visual search memory for each frame
        :v t
        :trace-detail high
        :act nil
        :bll 0.5
        :rt -2
        :ans 1
        :ol nil
        :seed nil ; override the fixed seed value from CogTool-KLM
        :er t
        )
)

(define-cogtool-model (:start-with-mouse t :timeout *CT-E_timeout* :initial-frame-name *CogTool-Explorer-start-frame*)

(chunk-type search-task
                        goal-text
                        best-cue ;;; holds widget text for facilitating trace output
                        best-loc
                        best-widget ;;; holds widget name for facilitating trace output
                        group
                        state
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; (p start) adds the goal chunk ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(p start
        =goal>
                isa             klm
                state           1
==>
```

```
        !bind! =goal-text *task-description*

        !eval! (reset-trial-parameters)
        !eval! (reset-frame-parameters)
        !eval! (transition *CogTool-Explorer-start-frame* *cogtool-design*)
        !eval! (if *CT-E-Debug* (print-visicon))

        +goal>
                isa             search-task
                goal-text       =goal-text
                best-cue        "dummy-text"
                best-loc        dummy-loc ;;;Will be updated to the first link assessed when model runs.
                best-widget     dummy-widget
                group           nil
                state           find
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Visual Search Productions ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(p find-unattended-in-frame-group
        =goal>
                isa                     search-task
                group                   nil
                state                   find

        ?visual-location>
                buffer                  empty

        ?visual>
                state                   free
==>
        +visual-location>
                isa                     cogtool-visual-location
                member-of-group         nil
                remote-label-of         nil ;;; Task specific knowledge to ignore Remote Labels
                - display-label         "Back" ;;; Task specific knowledge about Back button in browser window
                :attended               nil
                :nearest                current

        =goal>
                state                   finding
)

(p find-unattended-in-group
        =goal>
                isa                     search-task
                group                   =group
                state                   find
```

```
        ?visual-location>
                buffer                  empty

        ?visual>
                state                   free
==>
        +visual-location>
                isa                     cogtool-visual-location
                member-of-group         =group
                remote-label-of         nil ;;; Task specific knowledge to ignore Remote Labels
                - display-label         "Back" ;;; Task specific knowledge about Back button in browser window
                :attended               nil
                :nearest                current

        =goal>
                state                   finding
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Handle Visual Stuffing or Search Outcomes ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(p stuffed-with-unattended-in-frame-group
        =goal>
                isa                     search-task
                group                   nil
                state                   find

        =visual-location>
                isa                     cogtool-visual-location
                remote-label-of         nil ;;; Task specific knowledge to ignore Remote Labels
                - display-label         "Back" ;;; Task specific knowledge about Back button in browser window
                member-of-group         nil

        ?visual>
                state                   free
==>
        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object

        +visual>
                isa                     move-attention
                screen-pos              =visual-location

        =goal>
                state                   read-link
)

(p stuffed-with-unattended-in-group
        =goal>
                isa                     search-task
                group                   =group
```

```
               state                   find

        =visual-location>
               isa                     cogtool-visual-location
               remote-label-of         nil ;;; there Task specific knowledge to ignore Remote Labels
               - display-label         "Back" ;;; Task specific knowledge about Back button in browser window
               member-of-group         =group

        ?visual>
               state                   free
==>
        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object

        +visual>
               isa                     move-attention
               screen-pos              =visual-location

        =goal>
               state                   read-link
)

(p skip-stuffed-remote-label
        =goal>
               isa                     search-task
               state                   find

        =visual-location>
               isa                     cogtool-visual-location
               - remote-label-of       nil

        ?visual>
               state                   free
==>
        -visual-location>
)

(p skip-stuffed-back-button
        =goal>
               isa                     search-task
               state                   find

        =visual-location>
               isa                     cogtool-visual-location
               display-label           "Back" ;;; Task specific knowledge about Back button in browser window

        ?visual>
               state                   free
==>
        -visual-location>
)
```

```
(p skip-stuffed-unattended-in-wrong-group
       =goal>
              isa                     search-task
              group                   =group
              state                   find

       =visual-location>
              isa                     cogtool-visual-location
              - member-of-group       =group

       ?visual>
              state                   free
==>
       -visual-location>
)

(p skip-stuffed-unattended-in-group
       =goal>
              isa                     search-task
              group                   nil
              state                   find

       =visual-location>
              isa                     cogtool-visual-location
              - member-of-group       nil

       ?visual>
              state                   free
==>
       -visual-location>
)

(p no-more-unattended
       =goal>
              isa                     search-task
              - best-cue              "dummy-text"
              best-widget             =best-widget
              state                   finding

       ?visual-location>
              state                   error
==>
       !bind! =num-links-read (eval *number-widgets-read*) ; for debugging
       !output! (>>> Choose widget =best-widget after reading =num-links-read links] <<<) ; for debugging

       =goal>
              state                   choose-best-widget
)

(p no-unselected
       =goal>
```

```
               isa                     search-task
               best-cue                "dummy-text"
               state                   finding

       ?visual-location>
               state                   error
==>
       !bind! =state (if (= 0 (length *widget-infoscent-stack*)) 'stop 'go-back-to-previous)

       !eval! (if (and *update_confidence* (> (length *widget-infoscent-stack*) 1)) (push 0.01 (first (second *widget-infoscent-
stack*))))

       =goal>
               state                   =state
)

(p look-at-unattended
       =goal>
               isa                     search-task
               state                   finding

       =visual-location>
               isa                     cogtool-visual-location
               - display-label         "Back" ;;; Task specific knowledge about Back button in browser window

       ?visual>
               state                   free
==>
       =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object

       +visual>
               isa                     move-attention
               screen-pos              =visual-location

       =goal>
               state                   read-link
)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Read, check if is a chosen link and assess link ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(p target-link-not-on-this-page
       =goal>
               isa                     search-task
               state                   read-link

       =visual-location>
               isa                     cogtool-visual-location
               display-label           "Where the target link should be"
```

```
        =visual>
                isa                     visual-object
                screen-pos              =visual-location
==>
        !bind! =state (if (= 0 (length *widget-infoscent-stack*)) 'stop 'go-back-to-previous)

        !eval! (if (and *update_confidence* (> (length *widget-infoscent-stack*) 1)) (push 0.01 (first (second *widget-infoscent-
stack*))))

        =goal>
                state                   =state
)

(p read-widget
        =goal>
                isa                     search-task
                state                   read-link

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
                isa                     cogtool-visual-location
                - display-label         "Where the target link should be"
                textual-cue             =textual-cue
                widget-name             =widget-name

        =visual>
                isa                     visual-object
                screen-pos              =visual-location
==>
        !bind! =widget-name (if (and *handle-groups-in-half-flatten-layout* (search "Subordinate in" =widget-name))
                                        (subseq =widget-name 15)
                                        (if *recognize-equivalent-links-in-half-flatten-layout* =textual-cue =widget-name))

        +retrieval>
                isa                     visual-object
                value                   =widget-name
                status                  chosen

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object

        =visual> ; prevent implicit clearing of the visual buffer to maintain the chunk for modification in production click-mouse

        =goal>
                state                   assess-link
)

(p widget-was-previously-chosen
        !eval! (not *modified-previously-chosen-widgets*)

        =goal>
                isa                     search-task
                state                   assess-link
```

```
        ?retrieval>
                buffer                  full
==>
        -visual-location> ; so that find-unattended-* will fire, otherwise stuffed-* will fire over and over again

        -visual>

        =goal>
                state                   find
)

(p modified-widget-was-previously-chosen
        !eval! (eval *modified-previously-chosen-widgets*)

        =goal>
                isa                     search-task
                best-widget             =best-widget
                state                   assess-link

        ?retrieval>
                buffer                  full
==>
        !output! (>>> Assessing =best-widget [previously selected] <<<) ; for debugging

        !eval! (if *corrected-k-update* (incf *number-widgets-read*))

        !eval! (update-prod-parameters 0.01 *best-widget-scent*)

        =goal>
                state                   whats-next

        !bind! =num-widgets-read (if *corrected-k-update* (eval *number-widgets-read*) (incf *number-widgets-read*))
        !output! (>>> Best widget after assessing =num-widgets-read widgets is =best-widget <<<) ; for debugging
)

(p assess-widget
        =goal>
                isa                     search-task
                goal-text               =goal-text
                best-cue                =best-cue
                best-loc                =best-loc
                best-widget             =best-widget
                state                   assess-link

        ?retrieval>
                state                   error

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
                isa                     cogtool-visual-location
                textual-cue             =textual-cue
```

```
            widget-name             =widget-name
            display-label           =display-label

        =visual>
            isa                     visual-object
            screen-pos              =visual-location
==>
        !bind! =widget-name-or-display-label (if (equalp =display-label "") =widget-name =display-label)
        !output! (>>> Assessing =widget-name-or-display-label <<<) ; for debugging

        !eval! (if *corrected-k-update* (incf *number-widgets-read*))

        !bind! =new-best-cue (find-best-cue =goal-text =best-cue =textual-cue :update-prod t) ; compare inforscent values and update
production utilities
        !bind! =new-best-loc (if (equalp =new-best-cue =textual-cue) =visual-location =best-loc) ; update visual-location of best link
        !bind! =new-best-widget (if (equalp =new-best-cue =textual-cue) =widget-name-or-display-label =best-widget) ; update widget-
name of best link

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object

        =visual>

        =goal>
            best-cue                =new-best-cue
            best-loc                =new-best-loc
            best-widget             =new-best-widget
            state                   whats-next

        !bind! =num-links-read (if *corrected-k-update* (eval *number-widgets-read*) (incf *number-widgets-read*))
        !output! (>>> Best widget after assessing =num-links-read widgets is =new-best-widget <<<) ; for debugging
)

(spp assess-widget :at 0.275)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; The 3 competing productions as per SNIF-ACT 2 ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(p read-another-widget
        =goal>
            ISA                     search-task
            state                   whats-next
==>
        -visual-location> ; so that find-unattended-* will fire, otherwise stuffed-* will fire over and over again

        -visual> ;;; is this necessary?

        =goal>
            state                   find
)
```

```
(p choose-best-widget
       =goal>
              ISA                     search-task
              best-widget             =best-widget
              state                   whats-next

       =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
              isa                     cogtool-visual-location

       =visual>
              isa                     visual-object
==>
       !bind! =num-links-read (eval *number-widgets-read*) ; for debugging
       !output! (>>> Choosing best widget =best-widget after assessing =num-links-read widgets <<<)

       =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object

       =visual>

       =goal>
              state                   choose-best-widget
)

(p go-back
       =goal>
              ISA                     search-task
              state                   whats-next
==>
       !eval! (with-open-file (*file* (concatenate 'string
              (substitute #\- #\? (subseq *task-description* 0 9))
              "_C=" (write-to-string *go-back-cost*)
              "_K=" (write-to-string *SNIF-ACT-k-value*) ".txt")
              :direction :output
              :if-does-not-exist :create
              :if-exists :append)
              (format *file* (concatenate 'string (write-to-string (1+ *model-run*)) ";"
                                                                    (write-to-string *step-index*) ";"
                                                                    (if *current-group* *current-group* (name (curframe
*cogtool-design*))) ";"
                                                                    "GO-BACK" ";"
                                                                    "FIND" ";"
                                                                    (write-to-string (mp-time)) ";"
                                                                    (write-to-string *number-widgets-read*) "~%")))

       !eval! (incf *step-index*)

       -visual-location>

       =goal>
              state                   go-back-to-previous
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Productions for Motor actions after decision taken ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(p look-at-best-widget-if-not-already-looking-at-it
        =goal>
                ISA                     search-task
                best-widget             =best-widget
                best-loc                =best-loc
                state                   choose-best-widget

        ?visual>
                state                   free

        =visual>
                isa                     visual-object
          - screen-pos =best-loc
==>
        !output! (>>> Look at best widget =best-widget <<<)

        +visual-location> =best-loc

        +visual>
                isa                     move-attention
                screen-pos              =best-loc

        =goal>
                state                   choose-best-widget
)

(p look-at-best-widget-after-no-more-widgets-to-look-at
        =goal>
                ISA                     search-task
                best-widget             =best-widget
                best-loc                =best-loc
                state                   choose-best-widget

        ?visual>
                state                   free
                buffer                  empty ;;; apparently the visual buffer gets cleared when the request to visual-location fails
==>
        !output! (>>> Look at best widget =best-widget <<<)

        +visual-location> =best-loc

        +visual>
                isa                     move-attention
                screen-pos              =best-loc

        =goal>
```

```
               state                   choose-best-widget
)

(p change-focus-within-group
       =goal>
               ISA                     search-task
               best-loc                =best-loc
               group                   =group
               state                   choose-best-widget

       ?visual>
               state                   free

       =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
               isa                     cogtool-visual-location
               kind                    cogtool-group
               widget-name             =widget-name
               textual-cue             =textual-cue

       =visual>
               isa                     visual-object
               screen-pos              =best-loc
==>
       !output! (>>> Focus on best group =widget-name <<<)

       !eval! (push (list *best-widget-scent*) *widget-infoscent-list*) ;;; Apr 1 - To support *modified-mean-prev-page* and
*modified-mean-current-page*
       !eval! (push *widget-infoscent-list* *widget-infoscent-stack*)

       !eval! (with-open-file (*file* (concatenate 'string
               (substitute #\- #\? (subseq *task-description* 0 9))
               "_C=" (write-to-string *go-back-cost*)
               "_K=" (write-to-string *SNIF-ACT-k-value*) ".txt")
               :direction :output
               :if-does-not-exist :create
               :if-exists :append)
               (format *file* (concatenate 'string (write-to-string (1+ *model-run*)) ";"
                                                                                    (write-to-string *step-index*) ";"
                                                                                    =group ";"
                                                                                    (trim =widget-name) ";"
                                                                                    "FIND;"
                                                                                    (write-to-string (mp-time)) ";"
                                                                                    (write-to-string *number-widgets-read*) "~%")))

       !eval! (push (list (name (curframe *cogtool-design*)) =group) *history*)
       !eval! (incf *step-index*)

       !bind! =value (if *recognize-equivalent-links-in-half-flatten-layout* (subseq =widget-name 15) =widget-name)

       =visual>
               value                   =widget-name
```

```
               status                  chosen

       -visual> ; ensures modified chunk in visual buffer enters DM

       !eval! (reset-frame-parameters)
       !eval! (setf *current-group* =widget-name)

       =goal>
               best-cue                "dummy-text"
               best-loc                dummy-location
               group                   =widget-name
               state                   find
)

(p change-focus-within-frame-group
       =goal>
               ISA                     search-task
               best-loc                =best-loc
               group                   nil
               state                   choose-best-widget

       ?visual>
               state                   free

       =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
               isa                     cogtool-visual-location
               kind                    cogtool-group
               widget-name             =widget-name
               textual-cue             =textual-cue

       =visual>
               isa                     visual-object
;              screen-pos              =visual-location ; LHS fails to match if this line is un-commented
               screen-pos              =best-loc
==>
       !output! (>>> Focus on best group =widget-name <<<)

       !eval! (push (list *best-widget-scent*) *widget-infoscent-list*) ;;; To support *modified-mean-prev-page* and *modified-mean-
current-page*
       !eval! (push *widget-infoscent-list* *widget-infoscent-stack*)

       !eval! (with-open-file (*file* (concatenate 'string
               (substitute #\- #\? (subseq *task-description* 0 9))
               "_C=" (write-to-string *go-back-cost*)
               "_K=" (write-to-string *SNIF-ACT-k-value*) ".txt")
               :direction :output
               :if-does-not-exist :create
               :if-exists :append)
               (format *file* (concatenate 'string (write-to-string (1+ *model-run*)) ";"
                                                                                    (write-to-string *step-index*) ";"
                                                                                    (name (curframe *cogtool-design*)) ";"
```

```
                                                                    (trim =widget-name) ";"
                                                                    "FIND;"
                                                                    (write-to-string (mp-time)) ";"
                                                                    (write-to-string *number-widgets-read*) "~%")))

        !eval! (push (list (name (curframe *cogtool-design*)) nil) *history*)
        !eval! (incf *step-index*)

        !bind! =value (if *recognize-equivalent-links-in-half-flatten-layout* (subseq =widget-name 15) =widget-name)

        =visual>
                value                   =value
                status                  chosen

        -visual> ; ensures modified chunk in visual buffer enters DM

        !eval! (reset-frame-parameters)
        !eval! (setf *current-group* =widget-name)

        =goal>
                best-cue                "dummy-text"
                best-loc                dummy-location
                group                   =widget-name
                state                   find
)

(p move-mouse
        !eval! (not *use-finger-default-value*)

        =goal>
                ISA                     search-task
                best-loc                =best-loc
                state                   choose-best-widget

        ?visual>
                state                   free

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
                isa                     cogtool-visual-location
                - kind                  cogtool-group

        =visual>
                isa                     visual-object
                screen-pos              =best-loc

        ?manual>
                state                   free
==>
        +manual>
                isa                     move-cursor
                loc                     =visual-location
```

```
        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object

        =visual>

        =goal>
                state                   click
)

(p click-mouse-within-group
        =goal>
                ISA                     search-task
                best-loc                =best-loc
                group                   =group
                state                   click

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
                isa                     cogtool-visual-location
                widget-name             =widget-name
                textual-cue             =textual-cue

        =visual>
                isa                     visual-object
                screen-pos              =best-loc

        ?manual>
                state                   free
==>
        !bind! =frame-name (name (curframe *cogtool-design*))
        !eval! (push (list *best-widget-scent*) *widget-infoscent-list*) ;;; Apr 1 - To support *modified-mean-prev-page* and
*modified-mean-current-page*
        !eval! (push *widget-infoscent-list* *widget-infoscent-stack*)
        !eval! (push (list =frame-name =group) *history*)

        !bind! =widget-name (if *recognize-equivalent-links-in-half-flatten-layout* =textual-cue =widget-name)

        =visual>
                value                   =widget-name
                status                  chosen

        -visual> ; ensures modified chunk in visual buffer enters DM

        +manual>
                isa                     click-mouse

        =goal>
                state                   wait
)

(p click-mouse-within-frame_group
        =goal>
```

```
               ISA                     search-task
               best-loc                =best-loc
               group                   nil
               state                   click

       =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
               isa                     cogtool-visual-location
               widget-name             =widget-name
               textual-cue             =textual-cue

       =visual>
               isa                     visual-object
               screen-pos              =best-loc

       ?manual>
               state                   free
==>
       !bind! =frame-name (name (curframe *cogtool-design*))
       !eval! (push (list *best-widget-scent*) *widget-infoscent-list*) ;;; Apr 1 - To support *modified-mean-prev-page* and
*modified-mean-current-page*
       !eval! (push *widget-infoscent-list* *widget-infoscent-stack*)
       !eval! (push (list =frame-name nil) *history*)

       !bind! =widget-name (if *recognize-equivalent-links-in-half-flatten-layout* =textual-cue =widget-name)

       =visual>
               value                   =widget-name
               status                  chosen

       -visual> ; ensures modified chunk in visual buffer enters DM

       +manual>
               isa                     click-mouse

       =goal>
               state                   wait
)

(p tap-finger-within-group
       !eval! (eval *use-finger-default-value*)

       =goal>
               ISA                     search-task
               best-loc                =best-loc
               group                   =group
               state                   choose-best-widget

       ?visual>
               state                   free

       =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
```

```
                isa                     cogtool-visual-location
                - kind                  cogtool-group
                widget-name             =widget-name
                textual-cue             =textual-cue

        =visual>
                isa                     visual-object
                screen-pos              =best-loc

        ?manual>
                state                   free
==>
        !bind! =frame-name (name (curframe *cogtool-design*))
        !eval! (push (list *best-widget-scent*) *widget-infoscent-list*) ;;; Apr 1 - To support *modified-mean-prev-page* and
*modified-mean-current-page*
        !eval! (push *widget-infoscent-list* *widget-infoscent-stack*)
        !eval! (push (list =frame-name =group) *history*)

        !bind! =widget-name (if *recognize-equivalent-links-in-half-flatten-layout* =textual-cue =widget-name)

        =visual>
                value                   =widget-name
                status                  chosen

        -visual> ; ensures modified chunk in visual buffer enters DM

        +manual>
                isa                     move-cursor
                loc                     =visual-location

        =goal>
                state                   wait
)

(p tap-finger-within-frame-group
        !eval! (eval *use-finger-default-value*)

        =goal>
                ISA                     search-task
                best-loc                =best-loc
                group                   nil
                state                   choose-best-widget

        ?visual>
                state                   free

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
                isa                     cogtool-visual-location
                - kind                  cogtool-group
                widget-name             =widget-name
                textual-cue             =textual-cue
```

```
        =visual>
                isa                     visual-object
                screen-pos              =best-loc

        ?manual>
                state                   free
==>
        !bind! =frame-name (name (curframe *cogtool-design*))
        !eval! (push (list *best-widget-scent*) *widget-infoscent-list*) ;;; Apr 1 - To support *modified-mean-prev-page* and
*modified-mean-current-page*
        !eval! (push *widget-infoscent-list* *widget-infoscent-stack*)
        !eval! (push (list =frame-name nil) *history*)

        !bind! =widget-name (if *recognize-equivalent-links-in-half-flatten-layout* =textual-cue =widget-name)

        =visual>
                value                   =widget-name
                status                  chosen

        -visual> ; ensures modified chunk in visual buffer enters DM

        +manual>
                isa                     move-cursor
                loc                     =visual-location

        =goal>
                state                   wait
)


(p check-for-target-frame-within-frame-group
        =goal>
                ISA                     search-task
                best-widget             =best-widget
                group                   nil
                state                   wait

        ?manual>
                state                   free
==>
        !eval! (if *use-finger-default-value* (infer-transition *cogtool-design*)) ; CogTool device currently only call infer-
transition on a mouse click

        !bind! =state (if (equal (name (curframe *cogtool-design*)) (first *CogTool-Explorer-target-frames*)) 'found 'find) ;;; to be
modified to support multiple target frames

        ; update log file
        !eval! (with-open-file (*file* (concatenate 'string
                (substitute #\- #\? (subseq *task-description* 0 9))
                "_C=" (write-to-string *go-back-cost*)
```

```
            "_K=" (write-to-string *SNIF-ACT-k-value*) ".txt")
            :direction :output
            :if-does-not-exist :create
            :if-exists :append)
            (format *file* (concatenate 'string (write-to-string (1+ *model-run*)) ";"
                                                                    (write-to-string *step-index*) ";"
                                                                    (if (second (first *history*)) (second (first
*history*)) (first (first *history*))) ";"
                                                                    (trim =best-widget) ";"
                                                                    (string =state) ";"
                                                                    (write-to-string (mp-time)) ";"
                                                                    (write-to-string *number-widgets-read*) "~%")))

      !eval! (incf *step-index*)

      ; knowledge to handle a self-transition link as essentially null
      !eval! (if (equal (name (curframe *cogtool-design*)) (first (first *history*))) (progn (pop *history*) (pop *widget-infoscent-
stack*)))

      ; knowledge to focus on the subordinate group upon visiting a 2nd-level page
      !bind! =group (if (and *handle-groups-in-half-flatten-layout* (find (name (curframe *cogtool-design*)) *2nd-level-pages* :test
#'equal))
                        (progn (push (first *widget-infoscent-stack*) *widget-infoscent-stack*)
                               (push (list (name (curframe *cogtool-design*)) nil) *history*)
                                     (concatenate 'string "Subordinate in " (name (curframe *cogtool-design*))))
                               nil)
      !eval! (setf *current-group* =group)

      !eval! (reset-frame-parameters)

      =goal>
            best-cue              "dummy-text"
            best-loc              dummy-loc
            best-widget           dummy-widget
            group                 =group
            state                 =state
)

(p check-for-target-frame-within-group
      =goal>
            ISA                   search-task
            best-widget           =best-widget
            group                 =group
            state                 wait

      ?manual>
            state                 free
==>
      !eval! (if *use-finger-default-value* (infer-transition *cogtool-design*)) ; CogTool device currently only call infer-
transition on a mouse click
```

```
	!bind! =state (if (equal (name (curframe *cogtool-design*)) (first *CogTool-Explorer-target-frames*)) 'found 'find) ;;; to be
modified to support multiple target frames

	; update log file
	!eval! (with-open-file (*file* (concatenate 'string
		(substitute #\- #\? (subseq *task-description* 0 9))
		"_C=" (write-to-string *go-back-cost*)
		"_K=" (write-to-string *SNIF-ACT-k-value*) ".txt")
		:direction :output
		:if-does-not-exist :create
		:if-exists :append)
		(format *file* (concatenate 'string (write-to-string (1+ *model-run*)) ";"
                                                                    (write-to-string *step-index*) ";"
                                                                    (if (second (first *history*)) (second (first
*history*)) (first (first *history*))) ";"
                                                                    (trim =best-widget) ";"
                                                                    (string =state) ";"
                                                                    (write-to-string (mp-time)) ";"
                                                                    (write-to-string *number-widgets-read*) "~%")))

	!eval! (incf *step-index*)

	; "knowledge" to handle a self-transition link as essentially null
	!eval! (if (equal (name (curframe *cogtool-design*)) (first (first *history*))) (progn (pop *history*) (pop *widget-infoscent-
stack*)))
	!eval! (setf *current-group* (if (equal (name (curframe *cogtool-design*)) (first (first *history*))) =group nil))
	!bind! =group (if (equal (name (curframe *cogtool-design*)) (first (first *history*))) =group nil)

	!eval! (reset-frame-parameters)

	=goal>
		best-cue		"dummy-text"
		best-loc		dummy-loc
		best-widget		dummy-widget
		group			=group
		state			=state
)

(p go-back-to-previous-group
	=goal>
		isa			search-task
		- group			nil
		state			go-back-to-previous
==>
	!eval! (pop *widget-infoscent-stack*)
	!eval! (setf *current-group* (second (first *history*)))
	!bind! =group (second (pop *history*))

	!eval! (reset-frame-parameters)

	=goal>
```

```
                best-cue                "dummy-text"
                best-loc                dummy-loc
                best-widget             dummy-widget
                group                   =group
                state                   find
)

(p go-back-to-another-group-using-non-header-process
        !eval! (eval *mixed-visual-search*)

        =goal>
                isa                     search-task
                group                   =current-group
                state                   go-back-to-previous

        ?visual>
                state                   free
==>
        !eval! (setf *current-group* (second (first *history*)))
        !bind! =previous-group (second (first *history*))

        +visual-location>
                isa                     cogtool-visual-location
                kind                    cogtool-group
                - widget-name           =current-group
                member-of-group         =previous-group
                :nearest                current

        =goal>
                group                   =previous-group
                state                   non-header-process
)

(p look-at-another-group-in-non-header-process
        =goal>
                isa                     search-task
                state                   non-header-process

        =visual-location>
                isa                     cogtool-visual-location
                kind                    cogtool-group

        ?visual>
                state                   free
==>
        =visual-location>

        +visual>
                isa                     move-attention
                screen-pos              =visual-location
```

```
        =goal>
                state                   look-at-another-group
)

(p focus-on-another-group-in-non-header-process
        =goal>
                isa                     search-task
                group                   nil
                state                   look-at-another-group

        ?visual>
                state                   free

        =visual-location> ; temporary solution until cogtool-visual-location slots are moved to cogtool-visual-object
                isa                     cogtool-visual-location
                kind                    cogtool-group
                widget-name             =widget-name

        =visual>
                isa                     visual-object
==>
        !output! (>>> Focus on another group =widget-name <<<)

        !eval! (with-open-file (*file* (concatenate 'string
                (substitute #\- #\? (subseq *task-description* 0 9))
                "_C=" (write-to-string *go-back-cost*)
                "_K=" (write-to-string *SNIF-ACT-k-value*) ".txt")
                :direction :output
                :if-does-not-exist :create
                :if-exists :append)
                (format *file* (concatenate 'string (write-to-string (1+ *model-run*)) ";"
                                                                        (write-to-string *step-index*) ";"
                                                                        (name (curframe *cogtool-design*)) ";"
                                                                        (trim =widget-name) ";"
                                                                        "FIND;"
                                                                        (write-to-string (mp-time)) ";"
                                                                        (write-to-string 0) "~%")))

        !eval! (incf *step-index*)

        !eval! (update-infoscent-stack-in-non-header-process)
        !eval! (reset-frame-parameters)
        !eval! (setf *current-group* =widget-name)

        =goal>
                best-cue                "dummy-text"
                best-loc                dummy-location
                group                   =widget-name
                state                   find
)
```

```
(p go-back-to-previous-frame
       !eval! (eval *use-back-button*)

       =goal>
              isa                     search-task
              group                   nil
              state                   go-back-to-previous

       ?visual>
              state                   free
==>
       +visual-location>
              isa                     cogtool-visual-location
              display-label           "Back"

       =goal>
              state locate-back-button
)

(p look-at-back-button
       =goal>
              isa                     search-task
              state                   locate-back-button

       ?visual>
              state                   free

       =visual-location>
              isa                     cogtool-visual-location
              display-label           "Back"
==>
       =visual-location>

       +visual>
              isa                     move-attention
              screen-pos              =visual-location

       =goal>
              state                   look-at-back-button
)

(p move-mouse-to-back-button
       =goal>
              isa                     search-task
              state                   look-at-back-button

       ?visual>
              state                   free

       =visual-location>
              isa                     cogtool-visual-location
```

```
        =visual>
                isa                     visual-object

        ?manual>
                state                   free
==>
        +manual>
                isa                     move-cursor
                loc                     =visual-location

        =goal>
                state                   move-mouse-to-back-button
)

(p click-back-button
        =goal>
                isa                     search-task
                state                   move-mouse-to-back-button

        ?manual>
                state                   free
==>
        +manual>
                isa                     click-mouse

        =goal>
                state                   click-back-button
)

(p back-button-clicked
        =goal>
                isa                     search-task
                state                   click-back-button

        ?manual>
                state                   free
==>
        !eval! (pop *widget-infoscent-stack*)
        !eval! (setf *current-group* (second (first *history*)))
        !bind! =group (second (first *history*))

        !eval! (if *use-back-button-history-in-half-flatten-layout*
*history*))

        !eval! (reset-frame-parameters)

        =goal>
                best-cue                "dummy-text"
                best-loc                dummy-loc
                best-widget             dummy-widget
```

```
(transition (first (pop *history*)) *cogtool-design*) (pop
```

```
              group                   =group
              state                   find
)

(p go-back-to-previous-frame-direct
       !eval! (not *use-back-button*)

       =goal>
              isa                     search-task
              group                   nil
              state                   go-back-to-previous
==>
       !eval! (pop *widget-infoscent-stack*)
       !eval! (setf *current-group* (second (first *history*)))
       !bind! =group (second (first *history*))
       !eval! (transition (first (pop *history*)) *cogtool-design*)

       !eval! (reset-frame-parameters)

       =goal>
              best-cue                "dummy-text"
              best-loc                dummy-loc
              best-widget             dummy-widget
              group                   =group
              state                   find
)

);end of define-model

(dotimes (*model-run* (1- *number-of-runs*))
       (setq *cogtool-result* (cogtool-run-model))
       (format t "~A~%<<< CT-Explorer: between runs marker >>>" *cogtool-result*)
       (reset))

(setf *model-run* (1- *number-of-runs*))



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Source code from ct-explorer-support.lisp  ;;;
;;;                                            ;;;
;;; Includes source code from SNIF-ACT 2.0     ;;;
;;;    that was retained and modified          ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setf *widget-value-is-title* t) ;;LT
(setf *infer-transitions* t) ;;LT

(defparameter *code-trace* t)

(defparameter *update-prod-param-verbose* t)
```

```
(defun find-best-cue (query best-link link-text &key (update-prod t) (noise *infoscent-noise*))
  (progn
    (if *CT-E-Debug* (format t "~S " link-text))
    (setf *current-widget-scent* (profitability query (if (stringp link-text) link-text (car link-text)) noise nil))
    (if *CT-E-Debug* (format t "scent = ~S" *current-widget-scent*))

    (when (>= *current-widget-scent* *best-widget-scent*)
      (setf *best-widget-scent* *current-widget-scent*))
    (when update-prod
      (update-prod-parameters *current-widget-scent* *best-widget-scent*))
    (if (>= *current-widget-scent* *best-widget-scent*) link-text best-link)))

(defun profitability (query text &optional (noise *infoscent-noise*) (remove-stop-word-p t))
  (if (not (string= text "dummy-text"))
       (if *infoscent_variabilty_between_runs*
              (let ((prev-value (fetch-assessed-infoscent text)))
                     (if prev-value prev-value
                            (let ((new-value (link-activation query text noise remove-stop-word-p)))
                                   (store-assessed-infoscent text new-value) new-value)))
              (link-activation query text noise remove-stop-word-p))
    0.0))

(defun link-activation (query link &optional (noise *infoscent-noise*) (remove-stop-word-p t))
       (if *consider-familarity*
              (max 0.01 (+ (* (fetch-pmi query link) *similarity-scale-factor* (fetch-familarity link)) (act-r-noise noise)))
              (max 0.01 (+ (* (fetch-pmi query link) *similarity-scale-factor*) (act-r-noise noise)))))

(defun update-prod-parameters (cur-link-val best-link-val)
  (let ((read-value (car (car (no-output (spp-fct (list 'read-another-widget :u))))))
        (choose-value (car (car (no-output (spp-fct (list 'choose-best-widget :u))))))
        (go-back-value (car (car (no-output (spp-fct (list 'go-back :u)))))))
    (when (> cur-link-val  0) ;SNIF-ACT: read-another-widget may fire a lot otherwise ;CT-E: This is not be necessary anymore, as
infoscent values are not negative.

      (push cur-link-val *widget-infoscent-list*)
      (when (>= (length *widget-infoscent-stack*) 1)
        (if *CT-E-Debug* (format t ", go-back_formula_first_operand = ~S, go-back_formula_second_operand = ~S"
                       (if *second-best-prev-page* (second-best-previous-page (first *widget-infoscent-stack*)) (mean-previous-page
(first *widget-infoscent-stack*)))
                       (mean-current-page (first *widget-infoscent-stack*) *widget-infoscent-list*)))
        (spp-fct (list 'go-back :u (- (-
               (if *second-best-prev-page* (second-best-previous-page (first *widget-infoscent-stack*)) (mean-previous-page (first
*widget-infoscent-stack*)))
               (mean-current-page (first *widget-infoscent-stack*) *widget-infoscent-list*)
               ) *go-back-cost*))))

      (spp-fct (list 'read-another-widget :u (/ (+ read-value cur-link-val) (1+ *number-widgets-read*))))
      (spp-fct (list 'choose-best-widget :u (/ (+ choose-value best-link-val) (+ (1+ *number-widgets-read*) *SNIF-ACT-k-value*))))

      (when *update-prod-param-verbose*
```

```
        (spp read-another-widget :u)
        (spp choose-best-widget :u)
        (spp go-back :u))
      )))

(defun mean-previous-page (prev-page-list)
  (let ((numerator (if *modified-mean-prev-page* (- (apply #'+ (rest prev-page-list)) (first (last (first prev-page-list)))) (apply
#'+ (rest prev-page-list))))
        (denominator (if *modified-mean-prev-page* (- (length prev-page-list) 2) (- (length prev-page-list) 1))))
    (if (> denominator 0) (/ numerator denominator) 0)))

(defun second-best-previous-page (prev-page-list)
  (let ((temp-list (copy-seq (rest prev-page-list))))
    (second (sort temp-list #'>))))

(defun mean-current-page (prev-page-list curr-page-list)
       (if *modified-mean-current-page*
              (mean (append (first prev-page-list) curr-page-list))
              (if *update_confidence*
                     (mean (append (butlast (first prev-page-list)) curr-page-list))
                     (mean curr-page-list))))

(defun mean (num-list)
  (let ((sum (apply #'+ num-list))
        (len (length num-list)))
    (if (> len 0) (/ sum len) 0)))

(defun update-infoscent-stack-in-non-header-process ()
  (let* ((prev-page-list (first *widget-infoscent-stack*))
                (selected-infoscent (first (last (first prev-page-list))))
                (position-selected (position selected-infoscent prev-page-list))
                (mis-previous (mean-previous-page prev-page-list)))
       (progn
              (setf (nth position-selected (first *widget-infoscent-stack*)) (* mis-previous *non-header-process-conf-scale*))
              (setf (first (first *widget-infoscent-stack*)) (list (* mis-previous *non-header-process-conf-scale*))))))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Infoscent Storage              ;;;
;;;                                ;;;
;;; Should change "pmi" to "score" ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defvar *pmi* (make-hash-table :test #'equalp)) ;double hash table for pmi data

(defun fetch-pmi (word1 word2)
  (let ((h-table2 (gethash word1 *pmi*)))
    (if h-table2
        (gethash word2 h-table2)
      nil)))
```

```
(defun store-score (w1 w2 freq)
  (progn (store-pmi-fct w1 w2 freq)
         (store-pmi-fct w2 w1 freq)))

(defun store-pmi-fct (word1 word2 freq)
  (let ((h-table2 (gethash word1 *pmi*)))
    (if h-table2
        (setf (gethash word2 h-table2) freq)
      (progn
        (setf (gethash word1 *pmi*) (make-hash-table :test #'equalp))
        (setf (gethash word2 (gethash word1 *pmi*)) freq)))
    freq))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Overwrite find-current-locs-with-spec in ACT-R vision.lisp to ;;;
;;;    enable fluctuation factor when testing for nearest         ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defmethod find-current-locs-with-spec ((vis-mod vision-module) spec)
  "Assume that it's a valid visual-location chunk-spec with at most 1
   attended slot specification and one nearest spec"

  (let* ((main-spec (strip-request-parameters-from-chunk-spec spec))
         (attended (when (slot-in-chunk-spec-p spec :attended)
                     (car (chunk-spec-slot-spec spec :attended))))
         (slots (chunk-spec-slot-spec main-spec))
         (current (current-marker vis-mod))
         (current-type (when current (chunk-chunk-type-fct current)))
         (nearest (when (slot-in-chunk-spec-p spec :nearest)
                    (car (chunk-spec-slot-spec spec :nearest))))
         (min-max-tests nil))


    ;; Remap all current values to the current chunk

    (if current
        (dolist (x slots)
          (when (eq (third x) 'current)
            (if (find (second x) (chunk-type-slot-names-fct current-type))
                (setf (third x) (chunk-slot-value-fct current (second x)))
              (progn
                (print-warning "Current visual-location does not have a slot named ~S so it is ignored in the request."
                               (second x))
                (setf slots (remove x slots))))))
      (dolist (x slots)
        (when (eq (third x) 'current)
          (print-warning "There is no currently attended location.  So, request specifying ~S as current is being ignored."
                         (second x))
          (setf slots (remove x slots)))))

    ;; Remove all tests for highest and lowest for later
```

```
(dolist (x slots)
  (when (or (eq (third x) 'lowest)
            (eq (third x) 'highest))
    (push-last x min-max-tests)
    (setf slots (remove x slots))))

;; update the finsts an new markers

(when attended
  (update-new vis-mod)
  (check-finsts vis-mod))

;; find the chunks that match

(let ((possible-chunks (if attended
                           (matching-attended-chunks attended (visicon-chunks vis-mod t))
                         (visicon-chunks vis-mod t)))
      (changed nil))

  ;; Hack to reassign value slots as needed before testing

  (dolist (check possible-chunks)
    (when (chunk-real-visual-value check)
      (push (cons check (chunk-slot-value-fct check 'value)) changed)
      (fast-set-chunk-slot-value-fct check 'value (chunk-real-visual-value check))))

(let ((matching-chunks (find-matching-chunks (slot-specs-to-chunk-spec (chunk-spec-chunk-type main-spec) slots)
                                             :chunks possible-chunks :variable-char #\&)))
  ;; apply all of the lowest/highest constraints
  ;; in the order provided

  (dolist (x min-max-tests)
    (let ((value nil)
          (truth (first x))
          (slot (second x))
          (test (third x)))

      ;; find the min/max value
      (dolist (y matching-chunks)
        (let ((cur-val (fast-chunk-slot-value-fct y slot)))
          (unless (numberp cur-val)
            (setf value :fail)
            (print-warning "Cannot apply ~S constraint because not all chunks have a numerical value." x)
            (return))
          (when (or (null value)
                    (and (eq test 'lowest)
                         (< cur-val value))
                    (and (eq test 'highest)
                         (> cur-val value)))
            (setf value cur-val))))
```

```
    (setf matching-chunks (remove-if-not (lambda (z)
                                           (if (eq truth '=)
                                               (= value z)
                                             (not (= value z))))
                                         matching-chunks
                                         :key (lambda (z)
                                                (fast-chunk-slot-value-fct z slot))))))

;; if there's a nearest constraint then
;; apply that filter now

(when (and nearest matching-chunks)

  (if (or (eq (third nearest) 'current)
          (eq (third nearest) 'current-x)
          (eq (third nearest) 'current-y)
          (and (chunk-p-fct (third nearest))
               (chunk-type-subtype-p-fct (chunk-chunk-type-fct (third nearest)) 'visual-location)))

      (let ((value nil)
            (truth (first nearest))
            (test (third nearest))
            (matches nil)
            (current-loc (aif (current-marker vis-mod)
                              it
                              (progn
                                (model-warning "No location has yet been attended so current is assumed to be at 0,0.")
                                (car (define-chunks (isa visual-location screen-x 0 screen-y 0)))))))

        ;; find the min value
        (dolist (y matching-chunks)
          (let ((cur-val (cond ((eq test 'current)
                                (* (+ 1 (act-r-noise 0.1654)) (dist (xy-loc y) (xy-loc current-loc)))) ;;;;; Modified
                               ((eq test 'current-x)
                                (abs (- (fast-chunk-slot-value-fct y 'screen-x) (fast-chunk-slot-value-fct current-loc 'screen-x))))
                               ((eq test 'current-y)
                                (abs (- (fast-chunk-slot-value-fct y 'screen-y) (fast-chunk-slot-value-fct current-loc 'screen-y))))
                               (t
                                (dist (xy-loc y) (xy-loc test))))))
            (if (or (null value)
                    (< cur-val value))
                (progn
                  (setf value cur-val)
                  (setf matches (list y)))
              (when (= cur-val value)
                (push y matches)))))

        (setf matching-chunks matches))

      (progn
```

```
            (print-warning "Nearest test in a visual-location request must be current or a chunk that is a subtype of visual-
location.")
            (print-warning "Ignoring nearest request for ~S." (third nearest))))
          )

      ;; undo the value slots that were changed for matching purposes

      (dolist (restore changed)
        (fast-set-chunk-slot-value-fct (car restore) 'value (cdr restore)))

      matching-chunks))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Familarity and Assessed Infoscent ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defvar *familarity* (make-hash-table :test #'equalp))

(defun store-familarity (word value)
  (setf (gethash word *familarity*) value))

(defun fetch-familarity (word)
  (let ((value (gethash word *familarity*)))
       (if value value 1)))

(defun store-assessed-infoscent (text infoscent)
  (setf (gethash text *assessed-infoscent*) infoscent))

(defun fetch-assessed-infoscent (text)
  (gethash text *assessed-infoscent*))
```

D: Python Script to Process CogTool-Explorer 1.2 Log Files

```
import os

TRIAL, SEQ, PAGE, LINK, STATE, TIME, LOOKEDAT = (0, 1, 2, 3, 4, 5, 6)

linklevel_file = open('LinkLevel', 'w')
print >> linklevel_file, 'Task;Page;Link;Visits by M;Selections by M;Reselections by M;Consecutive Reselections by M;GB by
M;Terminations by M;Success by M'
# note: # visits to the top page = # link selections on the top page
# note: # visits to a subordinate page = # link selections on that subordinate page + go-backs from that subordinate page
# note: terminations on a header or subordinate page are not counted as visits to that page (to match the tabulation from participant
log files)
# note: selections of a header link to a page = go-backs from that page + terminations at leaf pages under that page

time_file = open('time', 'w')
print >> time_file, 'Task;Page;Avg Time Before Link Click by Model;Avg Links Looked-at Before Link Click by Model'

tasklevel_file = open('TaskLevel', 'w')
print >> tasklevel_file, 'Task;Avg Time Before Link Click in 2-click-success Trials by Model;No of Go-backs in All Trials by Model;#
1-C-S by M;' + \
      '# S by M;Avg C-to-S by M;Avg C-to-S on Top-L by M;Avg C-to-S on 2nd-L by M;Avg GB-to-S by M;' + \
      'Avg T-to-S by M;Avg C-to-C-T-to-S by M;' + \
      'Avg C-to-C-T-to-S on Top-L by M;Avg C-to-C-T-to-S on 2nd-L by M;' + \
      '# F by M;Avg Clicks in Failure Trials by Model;Avg Clicks on Headings in Failure Trials by Model;Avg Clicks on Sub Links in
Failure Trials by Model;Avg Go-backs in Failure Trials by Model;' + \
      'Avg Time in Failure Trials by Model;Avg Click-to-Click Time in Failure Trials by Model'

calibration_stats_file = open('calibrate_for_jmp', 'w')
print >> calibration_stats_file, 'Task;Page;Time to Click Link;Agent;Number of Links Looked-at'

names = os.listdir('.')
for name in names: # for each task
  if not name.endswith('txt'):
    continue
  in_file = open(name)
  visit_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion &
Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
  select_records = {
    '0 Start Page':{'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion & Philosophy':0, 'Physical
Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0},
    'Art, Language & Literature':{'National & Regional Literature':0, 'Literature & Writing':0, 'Architecture':0, 'Artists':0,
'Language':0, 'Writers & Poets':0, 'Decorative Arts':0, 'Legends & Folklore':0, 'National & Regional Art':0, 'Painting, Drawing, &
Graphic Arts':0, 'Sculpture':0, 'Periods & Styles':0, 'Photography':0, 'GO-BACK':0},
    'Life Science':{'Plants':0, 'People in Life Science':0, 'Medicine':0, 'Invertebrate Animals':0, 'Fish':0, 'Algae & Fungi':0,
'Agriculture, Foodstuffs, & Livestock':0, 'Mammals':0, 'Reptiles & Amphibians':0, 'Biological Principles & Concepts':0, 'Anatomy &
Physiology':0, 'Environment':0, 'Birds':0, 'Viruses, Monerans, & Protists':0, 'GO-BACK':0},
    'History':{'History of Asia & Australasia':0, 'People in European History':0, 'People in United States History':0, 'United States
History':0, 'African History':0, 'World History & Concepts':0, 'Ancient History':0, 'History of the Americas':0, 'European History':0,
'GO-BACK':0},
```

```
    'Geography':{'World Cities, Towns, & Villages':0, 'Regions of the World':0, 'Rivers, Lakes, & Waterways':0, 'Parks & Monuments':0,
'Countries':0, 'Canadian Provinces & Cities':0, 'Islands':0, 'Mountain Ranges, Peaks, & Landforms':0, 'U.S. Cities, Towns, &
Villages':0, 'Maps & Mapmaking':0, 'Oceans & Seas':0, 'Exploration & Explorers':0, 'U.S. States, Territories, & Regions':0, 'GO-
BACK':0},
    'Religion & Philosophy':{'Theology & Practices':0, 'Mythology':0, 'Religious Figures':0, 'Philosophy':0, 'Religions & Religious
Groups':0, 'Scripture':0, 'The Occult':0, 'GO-BACK':0},
    'Physical Science & Technology':{'Construction & Engineering':0, 'Chemistry':0, 'Earth Science':0, 'Computer Science &
Electronics':0, 'Machines & Tools':0, 'People in Physical Science':0, 'Astronomy & Space Science':0, 'Paleontology':0, 'Industry,
Mining, & Fuels':0, 'Physics':0, 'Transportation':0, 'Communications':0, 'Mathematics':0, 'Military Technology':0, 'Time, Weights, &
Measures':0, 'GO-BACK':0},
    'Social Science':{'Economics & Business':0, 'Organizations':0, 'Institutions':0, 'Political Science':0, 'Psychology':0, 'Law':0,
'Education':0, 'Anthropology':0, 'Military':0, 'Sociology & Social Reform':0, 'Calendar, Holidays, & Festivals':0, 'Archaeology':0,
'GO-BACK':0},
    'Sports, Hobbies, & Pets':{'Sports':0, 'Sports Figures':0, 'Games, Hobbies, & Recreation':0, 'Pets':0, 'GO-BACK':0},
    'Performing Arts':{'Theater':0, 'Musicians & Composers':0, 'Cinema, Television, & Broadcasting':0, 'Music':0, 'Dance':0, 'Musical
Instruments':0, 'GO-BACK':0}}
  page_terminate_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion
& Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

  leaf_terminate_records = {
    'Art, Language & Literature':{'National & Regional Literature':0, 'Literature & Writing':0, 'Architecture':0, 'Artists':0,
'Language':0, 'Writers & Poets':0, 'Decorative Arts':0, 'Legends & Folklore':0, 'National & Regional Art':0, 'Painting, Drawing, &
Graphic Arts':0, 'Sculpture':0, 'Periods & Styles':0, 'Photography':0, 'GO-BACK':0},
    'Life Science':{'Plants':0, 'People in Life Science':0, 'Medicine':0, 'Invertebrate Animals':0, 'Fish':0, 'Algae & Fungi':0,
'Agriculture, Foodstuffs, & Livestock':0, 'Mammals':0, 'Reptiles & Amphibians':0, 'Biological Principles & Concepts':0, 'Anatomy &
Physiology':0, 'Environment':0, 'Birds':0, 'Viruses, Monerans, & Protists':0, 'GO-BACK':0},
    'History':{'History of Asia & Australasia':0, 'People in European History':0, 'People in United States History':0, 'United States
History':0, 'African History':0, 'World History & Concepts':0, 'Ancient History':0, 'History of the Americas':0, 'European History':0,
'GO-BACK':0},
    'Geography':{'World Cities, Towns, & Villages':0, 'Regions of the World':0, 'Rivers, Lakes, & Waterways':0, 'Parks & Monuments':0,
'Countries':0, 'Canadian Provinces & Cities':0, 'Islands':0, 'Mountain Ranges, Peaks, & Landforms':0, 'U.S. Cities, Towns, &
Villages':0, 'Maps & Mapmaking':0, 'Oceans & Seas':0, 'Exploration & Explorers':0, 'U.S. States, Territories, & Regions':0, 'GO-
BACK':0},
    'Religion & Philosophy':{'Theology & Practices':0, 'Mythology':0, 'Religious Figures':0, 'Philosophy':0, 'Religions & Religious
Groups':0, 'Scripture':0, 'The Occult':0, 'GO-BACK':0},
    'Physical Science & Technology':{'Construction & Engineering':0, 'Chemistry':0, 'Earth Science':0, 'Computer Science &
Electronics':0, 'Machines & Tools':0, 'People in Physical Science':0, 'Astronomy & Space Science':0, 'Paleontology':0, 'Industry,
Mining, & Fuels':0, 'Physics':0, 'Transportation':0, 'Communications':0, 'Mathematics':0, 'Military Technology':0, 'Time, Weights, &
Measures':0, 'GO-BACK':0},
    'Social Science':{'Economics & Business':0, 'Organizations':0, 'Institutions':0, 'Political Science':0, 'Psychology':0, 'Law':0,
'Education':0, 'Anthropology':0, 'Military':0, 'Sociology & Social Reform':0, 'Calendar, Holidays, & Festivals':0, 'Archaeology':0,
'GO-BACK':0},
    'Sports, Hobbies, & Pets':{'Sports':0, 'Sports Figures':0, 'Games, Hobbies, & Recreation':0, 'Pets':0, 'GO-BACK':0},
    'Performing Arts':{'Theater':0, 'Musicians & Composers':0, 'Cinema, Television, & Broadcasting':0, 'Music':0, 'Dance':0, 'Musical
Instruments':0, 'GO-BACK':0}}

  successful_records = {
    'Art, Language & Literature':{'National & Regional Literature':0, 'Literature & Writing':0, 'Architecture':0, 'Artists':0,
'Language':0, 'Writers & Poets':0, 'Decorative Arts':0, 'Legends & Folklore':0, 'National & Regional Art':0, 'Painting, Drawing, &
Graphic Arts':0, 'Sculpture':0, 'Periods & Styles':0, 'Photography':0, 'GO-BACK':0},
```

```
    'Life Science':{'Plants':0, 'People in Life Science':0, 'Medicine':0, 'Invertebrate Animals':0, 'Fish':0, 'Algae & Fungi':0,
'Agriculture, Foodstuffs, & Livestock':0, 'Mammals':0, 'Reptiles & Amphibians':0, 'Biological Principles & Concepts':0, 'Anatomy &
Physiology':0, 'Environment':0, 'Birds':0, 'Viruses, Monerans, & Protists':0, 'GO-BACK':0},
    'History':{'History of Asia & Australasia':0, 'People in European History':0, 'People in United States History':0, 'United States
History':0, 'African History':0, 'World History & Concepts':0, 'Ancient History':0, 'History of the Americas':0, 'European History':0,
'GO-BACK':0},
    'Geography':{'World Cities, Towns, & Villages':0, 'Regions of the World':0, 'Rivers, Lakes, & Waterways':0, 'Parks & Monuments':0,
'Countries':0, 'Canadian Provinces & Cities':0, 'Islands':0, 'Mountain Ranges, Peaks, & Landforms':0, 'U.S. Cities, Towns, &
Villages':0, 'Maps & Mapmaking':0, 'Oceans & Seas':0, 'Exploration & Explorers':0, 'U.S. States, Territories, & Regions':0, 'GO-
BACK':0},
    'Religion & Philosophy':{'Theology & Practices':0, 'Mythology':0, 'Religious Figures':0, 'Philosophy':0, 'Religions & Religious
Groups':0, 'Scripture':0, 'The Occult':0, 'GO-BACK':0},
    'Physical Science & Technology':{'Construction & Engineering':0, 'Chemistry':0, 'Earth Science':0, 'Computer Science &
Electronics':0, 'Machines & Tools':0, 'People in Physical Science':0, 'Astronomy & Space Science':0, 'Paleontology':0, 'Industry,
Mining, & Fuels':0, 'Physics':0, 'Transportation':0, 'Communications':0, 'Mathematics':0, 'Military Technology':0, 'Time, Weights, &
Measures':0, 'GO-BACK':0},
    'Social Science':{'Economics & Business':0, 'Organizations':0, 'Institutions':0, 'Political Science':0, 'Psychology':0, 'Law':0,
'Education':0, 'Anthropology':0, 'Military':0, 'Sociology & Social Reform':0, 'Calendar, Holidays, & Festivals':0, 'Archaeology':0,
'GO-BACK':0},
    'Sports, Hobbies, & Pets':{'Sports':0, 'Sports Figures':0, 'Games, Hobbies, & Recreation':0, 'Pets':0, 'GO-BACK':0},
    'Performing Arts':{'Theater':0, 'Musicians & Composers':0, 'Cinema, Television, & Broadcasting':0, 'Music':0, 'Dance':0, 'Musical
Instruments':0, 'GO-BACK':0}}

  task_time_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion &
Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
  task_time_visit_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion
& Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

  trial_time_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion &
Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
  trial_time_visit_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0,
'Religion & Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

  task_look_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion &
Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

  trial_look_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion &
Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

  calibrate_time_records = {'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion & Philosophy':0,
'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

  calibrate_visit_records = {'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion & Philosophy':0,
'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

  first_select_records = {'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion & Philosophy':0,
'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
  reselect_records = {'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion & Philosophy':0,
'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
  consecutive_reselect_records = {'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion &
Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
```

```
prev_header = ''
prev_link = ''
prev_state = ''
trial_number = 0
prev_time_stamp = 0

success_count = 0
headings_clicks_success_trials = 0
link_clicks_success_trials = 0
go_back_count_success_trials = 0
time_success_trials = 0
heading_time_success_trials = 0
link_time_success_trials = 0

failure_count = 0
headings_clicks_failure_trials = 0
link_clicks_failure_trials = 0
go_back_count_failure_trials = 0
time_failure_trials = 0

go_back_count = 0
heading_clicks = 0
link_clicks = 0
heading_time = 0
link_time = 0
run_time = 0
prev_click_time = 0

while True: # looping through lines in the file
  line = in_file.readline()

  if line == '': # if EOF
    if prev_header != '' and prev_state == 'FIND':
      if prev_header != '0 Start Page':
        if prev_link != 'GO-BACK':
          leaf_terminate_records[prev_header][prev_link] = leaf_terminate_records[prev_header][prev_link] + 1
        else:
          visit_records['0 Start Page'] = visit_records['0 Start Page'] + 1
          page_terminate_records['0 Start Page'] = page_terminate_records['0 Start Page'] + 1
      else:
        visit_records[prev_link] = visit_records[prev_link] + 1
        page_terminate_records[prev_link] = page_terminate_records[prev_link] + 1
      failure_count = failure_count + 1
      go_back_count_failure_trials = go_back_count_failure_trials + go_back_count
      headings_clicks_failure_trials = headings_clicks_failure_trials + heading_clicks
      link_clicks_failure_trials = link_clicks_failure_trials + link_clicks

    # aggregate the time records for the last trial
    for page in task_time_records:
      if trial_time_visit_records[page] != 0:
```

```
            task_look_records[page] = task_look_records[page] + trial_look_records[page] / float(trial_time_visit_records[page])
            task_time_records[page] = task_time_records[page] + trial_time_records[page] / float(trial_time_visit_records[page])
            task_time_visit_records[page] = task_time_visit_records[page] + 1
        break

    tokens = line.split(';')
    # additional processing for grouped layouts
    if tokens[PAGE].startswith("Group ["):
      tokens[PAGE] = tokens[PAGE][7:len(tokens[PAGE])-17]
    if tokens[PAGE] == "Sports Hobbies Pets":
      tokens[PAGE] = "Sports, Hobbies, & Pets"
    if tokens[LINK].startswith("Group ["):
      tokens[LINK] = tokens[LINK][7:len(tokens[LINK])-17]
    elif tokens[LINK].endswith("in 0 Start Page"):
      tokens[LINK] = tokens[LINK][0:len(tokens[LINK])-16]
    if tokens[LINK] == "Sports Hobbies Pets":
      tokens[LINK] = "Sports, Hobbies, & Pets"

    # code to calibrate time (using only 2 click success trials)
    # above comment is superseeded; now used to track 1-click successes in Multi-Group Layout
    if link_clicks == 0 and tokens[STATE] == 'FOUND':
      calibrate_time_records[tokens[PAGE]] = calibrate_time_records[tokens[PAGE]] + float(tokens[TIME]) - prev_time_stamp
      calibrate_visit_records[tokens[PAGE]] = calibrate_visit_records[tokens[PAGE]] + 1
      calibration_stats_file.write(name[0:9] + ';' + tokens[PAGE] + ';' + str(float(tokens[TIME]) - prev_time_stamp) + ';Model;' +
tokens[LOOKEDAT])
      print >> calibration_stats_file, name[0:9] + ';Top;' + str(trial_time_records['0 Start Page']) + ';Model;' +
str(trial_look_records['0 Start Page'])

    # code to record time (only the 1st click on a top-level link and only 1st clicks on visits into a 2nd-level page
    if tokens[SEQ] == '1': # aggregate the time records for the previous trial
      for page in task_time_records:
        if trial_time_visit_records[page] != 0:
          task_look_records[page] = task_look_records[page] + trial_look_records[page] / float(trial_time_visit_records[page])
          task_time_records[page] = task_time_records[page] + trial_time_records[page] / float(trial_time_visit_records[page])
          task_time_visit_records[page] = task_time_visit_records[page] + 1
      trial_time_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion
& Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

      trial_time_visit_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0,
'Religion & Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
      trial_look_records = {'0 Start Page':0, 'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion
& Philosophy':0, 'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}

    if tokens[SEQ] == '1': # time for first click on a header link on the first visit to Top page
      prev_time_stamp = 0
      trial_time_records[tokens[PAGE]] = trial_time_records[tokens[PAGE]] + float(tokens[TIME]) - prev_time_stamp
      trial_time_visit_records[tokens[PAGE]] = trial_time_visit_records[tokens[PAGE]] + 1
      trial_look_records[tokens[PAGE]] = trial_look_records[tokens[PAGE]] + int(tokens[LOOKEDAT])
    if tokens[TRIAL] == trial_number and prev_header == '0 Start Page' and tokens[PAGE] == prev_link: # time for first click on a
subordinate link on a sequence of visits to a Subordinate page
      trial_time_records[tokens[PAGE]] = trial_time_records[tokens[PAGE]] + float(tokens[TIME]) - prev_time_stamp
```

```
      trial_time_visit_records[tokens[PAGE]] = trial_time_visit_records[tokens[PAGE]] + 1
      trial_look_records[tokens[PAGE]] = trial_look_records[tokens[PAGE]] + int(tokens[LOOKEDAT])

    if tokens[LINK] == tokens[PAGE]: # ignore selection the header link of its own page
      prev_time_stamp = float(tokens[TIME])
      prev_click_time = float(tokens[TIME])
      continue

    if tokens[TRIAL] == trial_number: # still in same trial
      if tokens[PAGE] == '0 Start Page' and prev_link != 'GO-BACK': # indicates a go-back after exhausting links on a subordinate page
        if prev_header != '0 Start Page':
          visit_records[prev_header] = visit_records[prev_header] + 1
          select_records[prev_header]['GO-BACK'] = select_records[prev_header]['GO-BACK'] + 1
          go_back_count = go_back_count + 1
        else: # case of 2 consecutive visits to Start Page
          visit_records[prev_link] = visit_records[prev_link] + 1
          select_records[prev_link]['GO-BACK'] = select_records[prev_link]['GO-BACK'] + 1
          go_back_count = go_back_count + 1
    else: # new trial from previous trial
      for link in first_select_records:
        if first_select_records[link] > 1:
          reselect_records[link] = reselect_records[link] + first_select_records[link] - 1
      first_select_records = {'Art, Language & Literature':0, 'Life Science':0, 'History':0, 'Geography':0, 'Religion & Philosophy':0,
'Physical Science & Technology':0, 'Social Science':0, 'Sports, Hobbies, & Pets':0, 'Performing Arts':0}
      if prev_header != '' and prev_state == 'FIND':
        if prev_header != '0 Start Page':
          if prev_link != 'GO-BACK':
            leaf_terminate_records[prev_header][prev_link] = leaf_terminate_records[prev_header][prev_link] + 1
          else:
            visit_records['0 Start Page'] = visit_records['0 Start Page'] + 1
            page_terminate_records['0 Start Page'] = page_terminate_records['0 Start Page'] + 1
        else:
          visit_records[prev_link] = visit_records[prev_link] + 1
          page_terminate_records[prev_link] = page_terminate_records[prev_link] + 1
        failure_count = failure_count + 1
        go_back_count_failure_trials = go_back_count_failure_trials + go_back_count
        headings_clicks_failure_trials = headings_clicks_failure_trials + heading_clicks
        link_clicks_failure_trials = link_clicks_failure_trials + link_clicks
        time_failure_trials = time_failure_trials + run_time
        go_back_count = 0
        heading_clicks = 0
        link_clicks = 0
        run_time = 0
        heading_time = 0
        link_time = 0
        prev_click_time = 0
        prev_header = ''

    visit_records[tokens[PAGE]] = visit_records[tokens[PAGE]] + 1
    select_records[tokens[PAGE]][tokens[LINK]] = select_records[tokens[PAGE]][tokens[LINK]] + 1
    if tokens[LINK] == 'GO-BACK':
```

```
      go_back_count = go_back_count + 1
    else:
      if tokens[PAGE] == '0 Start Page':
        heading_clicks = heading_clicks + 1
        heading_time = heading_time + float(tokens[TIME]) - prev_click_time
        first_select_records[tokens[LINK]] = first_select_records[tokens[LINK]] + 1
        if tokens[LINK] == prev_header:
          consecutive_reselect_records[tokens[LINK]] = consecutive_reselect_records[tokens[LINK]] + 1
      else:
        link_clicks = link_clicks + 1
        link_time = link_time + float(tokens[TIME]) - prev_click_time
      prev_click_time = float(tokens[TIME])
    run_time = run_time + float(tokens[TIME]) - prev_time_stamp
    if tokens[STATE] == 'FOUND': # This was a success trial
      leaf_terminate_records[tokens[PAGE]][tokens[LINK]] = leaf_terminate_records[tokens[PAGE]][tokens[LINK]] + 1
      successful_records[tokens[PAGE]][tokens[LINK]] = successful_records[tokens[PAGE]][tokens[LINK]] + 1
      success_count = success_count + 1
      go_back_count_success_trials = go_back_count_success_trials + go_back_count
      headings_clicks_success_trials = headings_clicks_success_trials + heading_clicks
      link_clicks_success_trials = link_clicks_success_trials + link_clicks
      heading_time_success_trials = heading_time_success_trials + heading_time
      link_time_success_trials = link_time_success_trials + link_time
      time_success_trials = time_success_trials + run_time
      go_back_count = 0
      heading_clicks = 0
      link_clicks = 0
      run_time = 0
      heading_time = 0
      link_time = 0
      prev_click_time = 0

    if tokens[STATE] == 'FOUND': # This was a success trial
      prev_header = ''
    else:
      prev_header = tokens[PAGE]
    prev_link = tokens[LINK]
    prev_state = tokens[STATE]
    trial_number = tokens[TRIAL]
    prev_time_stamp = float(tokens[TIME])

  for page in select_records:
    if page != '0 Start Page':
      print >> linklevel_file, name[0:9] + ';Top;' + page + ';' + str(visit_records['0 Start Page']-page_terminate_records['0 Start
Page']) + ';' + \
            str(select_records['0 Start Page'][page]) + ';' + str(reselect_records[page]) + ';' +
str(consecutive_reselect_records[page]) + ';' + \
            str(select_records[page]['GO-BACK']) + ';' + str(page_terminate_records[page])
      for link in select_records[page]:
        if link != 'GO-BACK': # this line added for easier combination of participant results with model results
          print >> linklevel_file, name[0:9] + ';' + page + ';' + link + ';' + str(visit_records[page]-page_terminate_records[page]) +
';' + \
```

```
                str(select_records[page][link]) + ';;;;' + str(leaf_terminate_records[page][link]) + ';' +
str(successful_records[page][link])
    # following two lines commented out for easier combination of participant results with model results
    #else:
    #  print >> linklevel_file, name[0:9] + ';' + page + ';;' + str(visit_records[page]-page_terminate_records[page]) + ';;;' +
str(page_terminate_records[page])
    if task_time_visit_records[page] != 0:
      print >> time_file, name[0:9] + ';' + page + ';' + str(task_time_records[page] / float(task_time_visit_records[page])) + ';' +
str(task_look_records[page] / float(task_time_visit_records[page]))
    else:
      print >> time_file, name[0:9] + ';' + page

  go_back_total = 0
  for page in select_records:
    if page != '0 Start Page':
      go_back_total = go_back_total + select_records[page]['GO-BACK']
  printed = False
  for page in calibrate_visit_records:
    if calibrate_visit_records[page] > 0:
      print >> tasklevel_file, name[0:9] + ';' + str(calibrate_time_records[page] / float(calibrate_visit_records[page])) + ';' +
str(go_back_total) + ';' \
            + str(calibrate_visit_records[page]) + ';' + str(success_count) + ';',
      if success_count != 0:
        print >> tasklevel_file, str(float(headings_clicks_success_trials + link_clicks_success_trials)/success_count) + ';' + \
              str(float(headings_clicks_success_trials)/success_count) + ';' + \
              str(float(link_clicks_success_trials)/success_count) + ';' + \
              str(float(go_back_count_success_trials)/success_count) + ';' + \
              str(float(time_success_trials)/success_count) + ';' + \
              str(float(time_success_trials)/(headings_clicks_success_trials + link_clicks_success_trials)) + ';' + \
              str(float(heading_time_success_trials)/headings_clicks_success_trials) + ';' + \
              str(float(link_time_success_trials)/link_clicks_success_trials) + ';',
      else:
        print >> tasklevel_file, ';;;;;;;;',
      print >> tasklevel_file, str(failure_count) + ';',
      if failure_count != 0:
        print >> tasklevel_file, str(float(headings_clicks_failure_trials + link_clicks_failure_trials)/failure_count) + ';' + \
              str(float(headings_clicks_failure_trials)/failure_count) + ';' + \
              str(float(link_clicks_failure_trials)/failure_count) + ';' + \
              str(float(go_back_count_failure_trials)/failure_count) + ';' + \
              str(float(time_failure_trials)/failure_count) + ';' + \
              str(float(time_failure_trials)/(headings_clicks_failure_trials + link_clicks_failure_trials))
      else:
        print >> tasklevel_file, ';;;;;'
      printed = True
  if not printed:
    print >> tasklevel_file, name[0:9] + ';;' + str(go_back_total) + ';0;' + str(success_count) + ';',
    if success_count != 0:
      print >> tasklevel_file, str(float(headings_clicks_success_trials + link_clicks_success_trials)/success_count) + ';' + \
            str(float(headings_clicks_success_trials)/success_count) + ';' + \
            str(float(link_clicks_success_trials)/success_count) + ';' + \
            str(float(go_back_count_success_trials)/success_count) + ';' + \
```

```
            str(float(time_success_trials)/success_count) + ';' + \
            str(float(time_success_trials)/(headings_clicks_success_trials + link_clicks_success_trials)) + ';' + \
            str(float(heading_time_success_trials)/headings_clicks_success_trials) + ';' + \
            str(float(link_time_success_trials)/link_clicks_success_trials) + ';',
    else:
      print >> tasklevel_file, ';;;;;;;;',
    print >> tasklevel_file, str(failure_count) + ';',
    if failure_count != 0:
      print >> tasklevel_file, str(float(headings_clicks_failure_trials + link_clicks_failure_trials)/failure_count) + ';' + \
            str(float(headings_clicks_failure_trials)/failure_count) + ';' + \
            str(float(link_clicks_failure_trials)/failure_count) + ';' + \
            str(float(go_back_count_failure_trials)/failure_count) + ';' + \
            str(float(time_failure_trials)/failure_count) + ';' + \
            str(float(time_failure_trials)/(headings_clicks_failure_trials + link_clicks_failure_trials))
    else:
      print >> tasklevel_file, ';;;;;'

  in_file.close()

linklevel_file.close()
time_file.close()
tasklevel_file.close()
calibration_stats_file.close()
```