ABSTRACT

GOODWIN, PRAIRIE ROSE. Error Recovery Microstrategies in a Touchscreen Environment. (Under the direction of Dr. Robert St. Amant.)

Touchscreens have seen widespread adoption in the last decade due to the rise of smartphones, tablets, and touchscreen laptops. New interaction techniques have emerged that match the capabilities of touch interfaces, with advantages for usability, but one problem has worsened: the probability that an error will occur in the basic task of choosing interface elements has significantly increased over point-and-click interactions. The limits of the human motor system have been documented extensively, and fingers are not precise input devices. Our approach is to develop cognitive models of users’ actions in this context. Computational modeling is a staple in cognitive science, an interdisciplinary field that integrates theories from psychology, physiology, and computation, in order to explore human cognition.

Researchers in human-computer interaction increasingly see the value of being able to predict user performance on a given task, by use of a model, before conducting user studies. Models derived from observed user behavior can also shed light on the cognitive strategies and microstrategies that may generalize to other domains, a topic of interest among cognitive scientists.

This dissertation makes three contributions. First, we present an extension to Cogulator, an open source cognitive modeling program, that allows for probabilistic branching and looping behavior in GOMS models. Second, we used that extension to identify and model the low level cognitive strategies that users apply in recovering from touch-screen target-selection errors. Cognitive modelers interested in touchscreen interactions can use this information to better simulate real human performance. Lastly, we developed interaction guidelines for visual feedback that can assist users in recovering more quickly.
Error Recovery Microstrategies in a Touchscreen Environment

by
Prairie Rose Goodwin

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2017

APPROVED BY:

__________________________  __________________________
Dr. David Roberts            Dr. Benjamin Watson

__________________________  __________________________
Dr. Christopher Healey       Dr. Robert St. Amant
Chair of Advisory Committee
DEDICATION

To all the women who came before me,
who fought the hard battles,
to give me the opportunities that brought me here.
BIOGRAPHY

Prairie Rose Goodwin grew up Rapid City, a small town in the Black Hills of South Dakota. She found her love of computing at Vassar College and received her Bachelor's in Computer Science in 2012 with minors in Mathematics and Greek and Roman Studies. Her passion for teaching led her to go directly from her undergraduate into the PhD program at North Carolina State University. There, she received her Master’s in Computer Science with a focus on Human-Computer Interaction in 2015. Her research, while always focused on users, has covered interface metaphors, computer-assisted tasks, and mobile development and usability, before finally focusing on cognitive modeling of error recovery.

During her time in graduate school, she was awarded the Dean’s Doctoral Fellowship. In addition, she received funding from NCSU while working as a Teacher’s Assistant for the Human-Computer Interaction classes as well as a Research Assistantship funded by the NSF. In between terms, she worked on the iOS team at a major mobile development company and held an adjunct faculty position at the Art Institute of Raleigh-Durham.
ACKNOWLEDGEMENTS

It takes a village. While there is only one name listed on this dissertation, it is the product of many minds.

I thank my committee for their time and support with special gratitude to my advisor, Rob St. Amant, for seeing this through to the end despite the challenges not being co-located. To Chris Healey with whom I worked in close proximity, and Dave Roberts and Ben Watson who were indispensable to getting me over the finish line.

To Jamee Bateau, thank you for lending your talents to me by making the assets for my experiments.

I want to thank my mentors at Vassar College, Marc Smith, Barry Jones, and Jenny Walter whose influence continues to shape the kind of person and professional I strive to be. I will carry your wisdom with me always.

To those that have been on this path with me, I thank you for the camaraderie with special acknowledgment to Adam Marrs, Sean Mealin, Chris Theisen, and Sandy Pogarcic.

I give special thanks to my mother, Susan Hayman who has helped me overcome so much in my life and to my father, Paul Hayman, who shaped my love of learning but never got to see me graduate from college. The most important thing you taught me was to frame every problem in terms of how can it be done instead of whether it can be done. To my sister, who has always provided infinite support, and to Marc Silling, who has been a positive force in our lives, thank you.

And finally, to my closest supporters, my family. To Dan Goodwin who has taken every step of this journey with me, who has supported me unconditionally, who has never been afraid to step up, I can't imagine anyone else by my side. Thank you. To my daughter Aria, who reminds me every day what is really important and whose wisdom I've needed on more than one occasion, thank you. And to my daughter Mirabel, whose mark is yet to be made, I love you.

I am grateful to all the others who have supported me as educators, friends, and mentors. I cannot name you all, but nonetheless, thank you.
## TABLE OF CONTENTS

**LIST OF TABLES** ........................................... vii

**LIST OF FIGURES** ................................................... ix

**Chapter 1  INTRODUCTION** ........................................... 1

  1.1 Situational Relevance ........................................... 1

  1.2 Importance of Cognitive Models ...................................... 4

  1.3 Dissertation Overview ........................................... 6

    1.3.1 Contributions ........................................... 6

    1.3.2 Structure of the thesis ...................................... 7

**Chapter 2  Related Work** ........................................... 8

  2.1 Touchscreen interaction design ....................................... 8

    2.1.1 Input Accuracy on touchscreens ................................ 10

    2.1.2 Solutions to mitigate errors in touch paradigm (Usability) 11

  2.2 Task Modeling in HCI ........................................... 12

    2.2.1 Engineering Models ........................................... 13

    2.2.2 Architectures ........................................... 14

  2.3 Cognitive Modeling User Interfaces .................................. 16

    2.3.1 Influential GOMS User Interfaces ................................ 17

    2.3.2 Cogulator ........................................... 18

  2.4 Human Performance Errors ........................................... 18

    2.4.1 Cognitive Slips ........................................... 19

    2.4.2 Motor Errors ........................................... 21

  2.5 Error Recovery ........................................... 22

  2.6 Strategies ........................................... 23

    2.6.1 Microstrategies ........................................... 24

**Chapter 3  Cogulator Extension** ...................................... 26

  3.1 Why Cogulator ........................................... 27

  3.2 Cogulator Features ........................................... 28

    3.2.1 Text-based interface for easy manipulation 28

    3.2.2 Working memory implementation ................................ 29

    3.2.3 Gantt Chart ........................................... 30

  3.3 Weaknesses ........................................... 31

    3.3.1 Only linear models ........................................... 31

    3.3.2 No operator enforcement ...................................... 32

  3.4 Extended functionality ........................................... 32

    3.4.1 Selection ........................................... 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.2 Stochastic Variation</td>
<td>35</td>
</tr>
<tr>
<td>Chapter 4 Cognitive Models of Microstrategies associated with Error Recovery</td>
<td>37</td>
</tr>
<tr>
<td>4.1 Instrumentation</td>
<td>37</td>
</tr>
<tr>
<td>4.1.1 Design Motivations</td>
<td>38</td>
</tr>
<tr>
<td>4.1.2 Hardware Apparatus</td>
<td>40</td>
</tr>
<tr>
<td>4.1.3 Eye-Tracking fixation identification algorithm</td>
<td>41</td>
</tr>
<tr>
<td>4.1.4 Instrumentation Study</td>
<td>44</td>
</tr>
<tr>
<td>4.2 User Studies</td>
<td>52</td>
</tr>
<tr>
<td>4.2.1 Pilot study: faux errors</td>
<td>52</td>
</tr>
<tr>
<td>4.2.2 Gnome Wars (Visual-Search Study)</td>
<td>62</td>
</tr>
<tr>
<td>4.2.3 Whack-a-Gnome (Visual feedback and recovery study)</td>
<td>73</td>
</tr>
<tr>
<td>4.3 Summary</td>
<td>97</td>
</tr>
<tr>
<td>Chapter 5 Guidelines for visual interaction</td>
<td>100</td>
</tr>
<tr>
<td>5.1 Benefits of improving error recovery</td>
<td>100</td>
</tr>
<tr>
<td>5.2 Existing mobile guidelines</td>
<td>101</td>
</tr>
<tr>
<td>5.3 Delivering feedback</td>
<td>102</td>
</tr>
<tr>
<td>5.3.1 Identifying user category</td>
<td>102</td>
</tr>
<tr>
<td>5.3.2 Guidelines</td>
<td>102</td>
</tr>
<tr>
<td>5.3.3 Discussion</td>
<td>105</td>
</tr>
<tr>
<td>Chapter 6 Conclusions</td>
<td>106</td>
</tr>
<tr>
<td>6.1 Summary</td>
<td>107</td>
</tr>
<tr>
<td>6.2 Limitations</td>
<td>108</td>
</tr>
<tr>
<td>6.3 Threats to validity</td>
<td>109</td>
</tr>
<tr>
<td>6.4 Future Work</td>
<td>110</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>111</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>123</td>
</tr>
<tr>
<td>Appendix A Appendix A</td>
<td>124</td>
</tr>
<tr>
<td>A.1 Calibration Bias Adjustment</td>
<td>126</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 4.1  Accuracy and Precision of fixations presented in visual degrees. Fixations were found for 1137 of 1200 targets with a total of 24,673 points of gaze data. The screen was on average $12.24^\circ$ wide and $6.12^\circ$ high, but varied with how far away the user was stood from the tablet. .................................. 47

Table 4.2  Rate of occurrence of Visual-Search separated by number of goal targets. ............................. 69

Table 4.3  Comparison of the results of the observed user data collected during the user study against the CPM-GOMS model of Visual-Search. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. No statistical differences were found for any input conditions. The model was run 500 times, and the results were aggregated to get a range of probabilistic behavior. ........................................... 72

Table 4.4  Breakdown of microstrategies in the second dimension of target selection observed during the baseline condition by user category and overall. .................................................... 84

Table 4.5  Average recovery times for careful users in conditions two and three which directly compared feedback being shown at the error location versus at the likely intended target. All t-tests are two-tailed assuming unequal variance. When feedback was shown at the error location, careful users showed significant improvement. This was not true for feedback shown to careful users at intended target. ........................................... 86

Table 4.6  Average recovery times for carefree users in the baseline and conditions four and five which directly compared showing feedback at the next logical action versus the next action and at the error location. All t-tests are two-tailed assuming unequal variance. When feedback was shown at all logical locations, carefree users showed significant improvement. This was not true for feedback shown to careful users at only the next logical action. ............................................................... 87

Table 4.7  Error-free target selection: comparison of the the observed user data collected during the user study against the CPM-GOMS model. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior. ...................................................... 90
| Table 4.8 | Careful user baseline model assuming one error and no visual feedback vs. observed user data comparison. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior. | 92 |
| Table 4.9 | Carefree user baseline model assuming one error and no visual feedback vs. observed user data comparison. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior. | 93 |
| Table 4.10 | Feedback processed through peripheral vision model vs. user data comparison. This included careful users with feedback shown at error location and carefree users with feedback shown at error location and next logical action. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior. | 95 |
| Table 4.11 | Feedback processed through peripheral vision model vs. user data comparison. All conditions not explicitly listed in the field of attention model are covered here. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior. | 96 |
LIST OF FIGURES

Figure 1.1 Multitasking behavior is achieved by interleaving tasks as required by time and accuracy constraints. Error recovery is separate from planned interleaving and can cause prolonged, unplanned periods of inattention. When multitasking in high-risk situations like driving, making even small errors can become dangerous. .................................................. 5

Figure 2.1 Shifting content from under a finger to a visible location can increase accuracy and mitigate issues stemming from finger occlusion. This approach has been integrated into standard on-screen mobile keyboards. ........................................................................................................... 9

Figure 3.1 Cogulator’s Interface with Gray and Boehm-Davis’s Figure 4. The sidebar contains all of the available operators as well as the ability to create new ones. Once the model’s text has been processed, a Gantt Chart is available in a pull up tab at the bottom of the window. The time estimation can be seen and at the end of the chart, and at the top of the window next to the model name. A visualization of working memory load is below the Gantt chart. Goals are labeled on the Gantt chart just above working memory. ................................................................. 28

Figure 3.2 Figure 4 from Gray and Boehm-Davis “Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior”. Original caption: The location of TARGET is not known until it is perceived and verified. The CPM-GOMS prediction for time from mouse down on HOME to mouse down on TARGET is 970 msec. The critical path is indicated by bold lines connecting shadowed boxes. ............................................................................................................. 30

Figure 3.3 The Gantt chart visualizing the same task as Figure 3.2. Automatically generated by Cogulator. The critical path is a single unbroken line from start to finish. Multitasking operators not on the critical path can be seen as disconnected lines. ................................................................................. 31

Figure 3.4 New operators integrated into sidebar for easy access ..................... 34

Figure 3.5 New interface elements were added to report aggregate completion time data. The modeler enters the number of times the model should be run in a textbox and presses the associated button. Labels with the average, minimum, and maximum time appear to the right of the button. 35
Figure 4.1  Experimental Setup: The EyeTribe eye-tracker attached to the front of the Surface Pro 2 tablet along the bottom edge using the EyeTribe's proprietary harness and connected with a short cord in the USB on the left side. Both devices were attached to an adjustable height tripod with a universal 10" tablet tripod mount. If necessary, a wireless Bluetooth mouse was placed to the right of the tripod at a comfortable height for the standing participants. 41

Figure 4.2  The crosshair and circle targets used during the instrumentation study. 45

Figure 4.3  A boxplot with the means identified by the line in the center of the box, and the first and third quartile as the edges. The whiskers are placed at the 5% and 95% markers. Each data point was created by calculating the distance between the center point of the fixation and the center point of the target's true location in visual degrees. 47

Figure 4.4  A randomized target from the instrumentation study with eye-tracking data superimposed. The large circle was the actual size and location of the target. The small dots represent every gaze point collected while the target was visible. The dark cluster of dots are the gaze points that our algorithm identified as the fixation. 48

Figure 4.5  Confidence intervals can be associated with the identified eye-tracking data based on the percentage of the data that is within a certain radius from true target location. 50% of all fixations were within 1.12° or less, and 95% of fixations are within 2.75° or less. 49

Figure 4.6  Sample round that participants completed for the pilot study. When successfully touched, the red target would turn grey. Once all targets had been successfully touched, the round ended. 55

Figure 4.7  Visualization made to browse the order of events. A timeline can be seen in the bottom-left of the screen. Ovals represent the order in which the targets were activated. The gray circle represents a fixation, and lines connect them on the screen to show the order. A square represents a touch event, and it is overlaid on the fixation active immediately before the touch. 56

Figure 4.8  Bubble plot showing the perceived error rate versus the modified error rate. Radius is determined by frequency. Black trend line represents a linear regression model($r^2 = 0.5774$). The Pearson product-moment correlation coefficient between the two variables is 0.7498. 58

Figure 4.9  Visualizations of six microstrategies along a timeline separated by perceptual and motor processors. Ovals represent fixations, with color uniquely mapping to a target. Squares represent touches. The top color indicates success: green is successful, gray is unsuccessful. The bottom color follows the same color scheme as fixations. 59
Figure 4.10  The two gnome assets in the user studies. The happy gnome (left) functioned as a distractor. The angry gnome (right) functioned as the goal target that the participant had to locate. Color pallets, shape, and sizes were held constant to prevent this task being completed with peripheral vision. The background circle was added to give a visual cue of the active area and to normalize the shape. ..............................................

Figure 4.11  Sample level of the Gnome Wars game used to study Visual-Search. Users had to tap all angry gnomes to make them disappear. Once all enemies were gone, the level ended. ......................................................

Figure 4.12  Visualized results of partially automated procedure for correcting calibration errors post-collection. The unprocessed data (left) has fixations below and to the left of the true location of targets. After the same translation is applied to all data points by the minimum-distance heuristic(right), the Point-Of-Interest of each fixation is unambiguous. Turquoise circles are interface elements (level, score, and timer), the blue circles are distractors, and the red circles are goal targets. Fixations are represented as transparent ovals connected by lines in chronological order. .................................................................

Figure 4.13  Distributions of all durations of Visual-Search. No statistical differences were found for any input conditions. Model was run 500 times and the data was aggregated. .................................................................

Figure 4.14  One possible run of the Visual-Search CPM-GOMS model with stochastic variability. .................................................................

Figure 4.15  Sample level of the Whack-a-Gnome game used to study how different kinds of visual feedback affect error recovery. ......................................................

Figure 4.16  Feedback assets used to provide visual feedback when errors occurred. ......................................................

Figure 4.17  D2: No-Visual-Feedback microstrategy depicted in Gantt chart generated by Cogulator. In this example, the touch event starts before the fixation ends. Another form of NVF has the touch event starting strictly after the fixation is over. ......................................................

Figure 4.18  D2: With-Visual-Feedback microstrategy depicted in Gantt chart generated by Cogulator. Users verify the success or failure of their action before shifting their attention to the next action. ......................................................

Figure 4.19  Peripheral-Focus microstrategy depicted in Gantt chart generated by Cogulator. This microstrategy cannot be broken down by dimension because there is no clear line when one dimension ends and the next begins when this microstrategy is used. ......................................................

Figure 4.20  D3: Delayed-Error-Recovery depicted in Gantt chart generated by Cogulator. The user makes an error, notices it, but moves on to the next action before returning to recover from this error. ......................................................
Figure 4.21  D3: *Immediate-Error-Recovery* microstrategy depicted in Gantt chart generated by Cogulator. The user notices his or her error and recovers from it before continuing. ................................................. 82

Figure 4.22  D3: *Peck* microstrategy depicted in Gantt chart generated by Cogulator. The user continuously taps a button without cognitive pause. ................. 83

Figure 4.23  D3: *Muscle Memory* microstrategy depicted in Gantt chart generated by Cogulator. The user does not process actions cognitively in between actions. ................................................................. 83

Figure 4.24  Distributions of completion times assuming error-free performance for model and user data. Model was run 1000 times to get a range of realistic performance. .................................................. 90

Figure 4.25  Careful user baseline model distribution assuming one error and no visual feedback vs. observed user data distribution. Model was run 1000 times to get a range of probabilistic behavior. .................. 91

Figure 4.26  Carefree user baseline model distribution assuming one error and no visual feedback vs. observed user data distribution. Model was run 1000 times to get a range of probabilistic behavior. .............. 93

Figure 4.27  Feedback processed through peripheral vision model vs. user data distribution comparison. This included careful users with feedback shown at error location and carefree users with feedback shown at error location and next logical action. Model was run 1000 times to get a range of realistic completion times. .............................................. 94

Figure 4.28  Feedback processed through peripheral vision model vs. user data distribution comparison. All conditions not explicitly listed in the field of attention model are covered here. Model was run 1000 times to get a range of realistic completion times. ......................... 96
1.1 Situational Relevance

Touchscreens are becoming ubiquitous in the modern computing and mobile spheres. E.A. Johnson introduced the first touchscreen in 1965, but it took several decades for the technology to become widely used in consumer devices. Now, touchscreens are in phones, PC’s, ATM’s, cars, video game consoles, and all kinds of other consumer electronics. With nearly two billion smartphone users worldwide, understanding interaction differences between touchscreens and mouse-driven interfaces is imperative.

Touchscreens were a natural evolution in mobile computing because they removed the need for peripheral devices such as keyboards and navigation input (mouse, trackpad, eraser nub, etc.) In 1993, IBM released Simon, the first touchscreen phone. The resistive touchscreen was operated with a stylus, and the feature appeared in many luxury PDAs over the next decade. The first smartphones started appearing in consumer markets in the
early 2000’s. These devices were defined by their ability to access the Internet and run a variety of useful applications (later dubbed “apps”). Blackberry, Palm, and Windows mobile all made handsets that were considered to be high end luxury and business devices. In particular, Blackberry’s encryption standards were attractive to government agencies and businesses alike, and for years Blackberry was the leader in mobile technology [HK09].

When the iPhone launched in June 2007, the mobile computing landscape was transformed over night. Unlike Blackberry and Palm, which employed a stylus and small physical keyboards or number pads, the iPhone was made to be operated exclusively through finger-centric touch interactions. It was not the first capacitive touchscreen phone (LG Prada launched in May, 2007), but Apple’s attention to user experience and design made it an industry leader immediately. Blackberry and Palm failed to adapt and were replaced by iPhone, Android, and minorly successful OS’s including Windows Phone, Amazon Fire, and Mozilla. The iPhone 8, 8 plus, and iPhone X (10th anniversary edition), released in September 2017 mark Apple’s 16th, 17th, and 18th handset models. All together, Apple holds 13% of the market share. The Android ecosystem is fragmented with 24,000 models across 1,000 manufacturers worldwide [Gil15], but all together they hold 85% of the market. Windows Phone, the third most popular smartphone operating system holds less than 2% of the market share despite merging with established manufacturers [eri15]. There are now more than two billion smartphone users worldwide, and that number is expected to rise to six billion users by 2020 [eri15].

In addition to smartphones, touchscreens are being integrated with mouse-driven environments through laptops and desktops. Hardware manufacturers started introducing touchscreen capabilities into high-end laptops when the technology was becoming popular in smartphones. However, the user experience of these models lagged behind the needs of the consumer. Often, the touchscreen proved to be slow, inaccurate, and clearly added as an afterthought onto interfaces that were designed for point-and-click. Windows 8 was the first desktop operating system that tried to integrate touch-centric UI design into a traditional PC interface. Microsoft’s goal was to increase touchscreen adoption rates by unifying the experience across Windows Phone, tablet, laptop, and desktop. By 2014 when the first major update for Windows 8 was released, 13% of laptops had an integrated touchscreen [Trene]. Despite increasing adoption rates, users were critical of Windows 8’s interface design as soon
as it launched. Some of the design choices that they made to support touch interactions in desktop interfaces were actually detrimental to point-and-click interactions that still dominated the PC world. The next Windows operating system, Windows 10, reverted some of the radical UI changes (e.g. bringing back a corner located start menu), and the changes have been well received by consumers [Cha15].

Input accuracy continues to be the most critical problem in touchscreen usability. The cost of errors is usually small but can escalate given the circumstances. The smallest cost of an error is time wasted as the user must recognize that an error has been made and recover from it. If the user misses a button, it is not difficult to repeat the action. However, the opposite error has a much higher cost. If the user touches a button accidentally, the action must first be reversed before the correction can be made. In most settings, this can be accomplished easily with a back button or other similar “undo” features. In typing, several actions may have to be deleted in order to get to the mistake, increasing the time cost of the original error. If the error goes unnoticed, there could be a social cost of miscommunication. For example, auto-correct can completely change the intended meaning of a message. If an action cannot be reversed, then the cost of the error becomes significantly higher. Unintentionally sending messages and deleting information can have varying costs, while tapping the wrong button during a transaction can even result in monetary consequences. Often users perform tasks on mobile devices concurrently with other tasks like walking or driving. An error on one task can result in focus away from the other which can have very real and life threatening costs. To fix an error, the original context must be maintained which requires a high cognitive load. Users defer context switching until a time of low cognitive workload [SB10]; thus, the time cost of fixing an error is at the expense of the other activity.

Current research in touchscreen errors tends to focus on reducing the cost or frequency of their occurrence. There are currently two categories of solutions: those that try to keep an error from happening [PH10] [Par06], and those that compensate for the error once it occurs [Goe12] [Goe13]. These roughly correspond to design-time and run-time techniques for error identification and recovery. Neither approach can eliminate the occurrence of errors. Instead, we must accept that errors are inherent in the technology, and focus on understanding how they affect use.
1.2 Importance of Cognitive Models

Card, Moran, and Newell [Car83] specified three engineering models to estimate user performance, each focusing on a different facet of a task. The Keystroke-Level Model is based on low-level operators for actions such as pressing keys and pointing. GOMS (Goals, Operators, Methods, and Selection Rules) is based on sequential and hierarchical tasks. The Model Human Processor is based on an information processing model of cognition with separate processors for motor, perceptual, and cognitive actions. Over the years, information processing models have matured into detailed cognitive architectures like ACT-R [And96], EPIC [KM97], and Soar [Lai87]. These frameworks are able to formalize usability of interfaces and have been shown to be a good predictor of user performance. Moreover, cognitive models can begin to explain why certain aspects of an interface are problematic.

Cognitive modeling when combined with biometric data such as eye-tracking can identify the cognitive strategies used to complete a task [SA01]. Eye-tracking provides an unobtrusive form of data collection that allows researchers to observe natural behavior [MR02] [Hut89]. Psychology research has been able to use eye-tracking to pinpoint search strategies, problem solving, focus shifts, and other thought processes [Bal03]. Points of interest (PoIs) can be identified with eye-tracking, and the cognitive model can shed light on the strategies that correspond to the data [Byr01]. Once modelers understand how users interact with their system, design changes can be made to improve their experience. While the technology has been around for decades, improvements in accuracy and affordability of eye-trackers have recently made it an attractive option for researchers.

Project Ernestine demonstrated that cognitive modeling can accurately predict expert performance without the need to train participants and evaluate their performance on a new system [Gra93]. In the early 1990’s, a telephone operator company designed new workstations to reduce the time that operators spent on each call. Every additional second cost the company $3 million over the course of a year. The work stations were evaluated using GOMS cognitive models and a long-term user study. Contrary to expectations, the GOMS model predicted an increase in call time that was verified later by the user data. Due to time and resource constraints most usability studies do not have the ability to prototype, develop, and train users up to peak performance. Instead, they are often short
Multitasking behavior is achieved by interleaving tasks as required by time and accuracy constraints. Error recovery is separate from planned interleaving and can cause prolonged, unplanned periods of inattention. When multitasking in high-risk situations like driving, making even small errors can become dangerous.

Term experiments that do not capture familiarity with a product. This kind of evaluation is sufficient for instances in which a user may only interact with product for short periods of time but are not reliable representations of long term usage. In many cases, cognitive modeling may be the only viable option for estimating long term gains.

Our work models very low-level unconscious behavior known as microstrategies that are exclusive to the process of error recovery (see Figure 1.1). Between motor, visual, and cognitive activities, there are a limited number of observable events in a touch interaction. The visual events are start of a fixation, end of a fixation, and a saccade. The motor events are move hand to position, touch-down, and touch-up. The cognitive event is a less defined “processing” event that occurs during the pauses necessary for the user to react, and the purpose can be inferred from the surrounding events. Recurring patterns of these events comprise microstrategies that underpin observable behavior [GBD00]. Microstrategies are chosen to complete one step in a task and can change based on subtle features of an interface or goals. This work discusses how visual feedback changes the microstrategies associated with error recovery.
1.3 Dissertation Overview

When an error is made in a touchscreen environment, visual feedback can be tailored to the user in order to assist in error recovery. Target selection errors on touchscreens are an inherent issue due to the mismatch between the requirements of the technology and accuracy limitations of the human motor system. With this in mind, this dissertation asks two research questions:

• Q1: What microstrategies do people use when recovering from a target selection error during touchscreen interactions?

• Q2: How can we change touchscreen interfaces to reduce the time necessary to recover from the error?

To answer these questions, we performed multiple rounds of user studies. We examined the microstrategies that people used in target selection and created models of error recovery in CPM-GOMS. In addition, we present an extension to Cogulator, a cognitive modeling framework. Our modeling efforts are the first to formalize a set of domain-independent models of error recovery for target selection.

1.3.1 Contributions

The answers to our research questions have both theoretical and practical benefits. As such, this dissertation is organized around three distinct contributions:

• An extension to Cogulator that allows modelers to include ifs, states, and gotos. These new operators increase the utility of the language by adding the ability to write branches, loops, jumps, and conditionals, significantly expanding the types of models that Cogulator can handle. Additionally, our extension gives the modeler the ability to use stochastic branching to get a range of realistic completion times.

• A set of domain-independent models of target selection and error recovery strategies. Error recovery is largely ignored by the modeling community, and common errors and recovery strategies have not been fully explored. These models can be used as secondary tasks to better predict user performance on touchscreen devices.
• Guidelines for presenting visual feedback on a touchscreen device that minimize
the time necessary to recover from errors. Visual feedback is not delivered effectively
on mobile devices. This results in users not noticing that they have made an error,
compounding the time and potential costs. Our work informs designers how to
determine what kind of user they are working with and how to tailor to their needs.

1.3.2 Structure of the thesis

Chapter 2 presents all of the background information necessary to explain the importance
of this work. We start with a brief overview of touchscreen interactions for context and
current cognitive modeling frameworks and tools. We then discuss how errors are currently
modeled and examine how strategies can be informative to analyzing cognitive processes.

Chapter 3 discusses the Cogulator extension that we wrote for our modeling purposes.
It describes the strengths and weaknesses of Cogulator and makes our case for why we
chose to use it for this work. We then go into the technical details of how we modified it
and the features that we added.

Chapter 4 presents the microstrategies that we observed and predicted. After running
pilot studies that laid the ground work for our experimental setup, we designed a two-part
experiment. The first experiment identifies two classes of users: those who use careful
microstrategies, and those who do not. The second experiment predicted how changing
the visual feedback changes the microstrategies employed by each class of users. We then
provide an in-depth breakdown of the task and present the cognitive models of each step.

Chapter 5 condenses the results from our experiments into an actionable guidelines for
visual feedback to assist users in error recovery. We discuss methodologies for identifying
which class the user is in, what factors may affect error recovery strategies, and where to
present visual feedback.

Finally, chapter 6 provides a discussion for the overall contributions of this work. Here,
we discuss threats to validity and potential directions for future work.
2.1 Touchscreen interaction design

The mobile nature of touchscreens provide new challenges that did not exist in mouse-driven environments. Hardware constraints force developers to consider data usage, slow transfer speeds, limited computational power, and a finite battery. The tradeoff becomes one of relying on network stability to offload tasks to the cloud or using local resources with reduced capabilities. This balancing act often results in subtle changes that can greatly affect users’ experience with a product.

The small screen significantly limits screen-real estate, requiring information to be extremely concise, and features to be incorporated very carefully. Research has shown that people’s ability to retain information read on a mobile screen is directly correlated to its complexity. Moreover, mobile device users comprehend less than half (48%) as much information as individuals using a desktop [Nie11] [Mey16].
Beyond hardware constraints, touch interactions provide new challenges that require their own design considerations. The most obvious of these is finger occlusion [BC09]. By necessity, users cannot see the object on which they are tapping. Interface solutions to finger occlusion include increasing the size of targets (taking up even more limited screen-real estate) or displaying what is under the user's finger in a shifted window [VB07] as shown in Figure 2.1. Both solutions have been adopted into standard mobile interfaces. The shifted window approach can also assist with the problem of multi-pixel activation. The finger pad is much larger than the tip of a cursor, which introduces ambiguity about the exact activation point. The window provides visual feedback to the user about which pixel is being activated, and this approach is the standard for text-selection and typing on digital keyboards.

![Figure 2.1](image)

**Figure 2.1** Shifting content from under a finger to a visible location can increase accuracy and mitigate issues stemming from finger occlusion. This approach has been integrated into standard on-screen mobile keyboards.

In order to solve some of the problems of reduced screen availability, designers have experimented with gesture-based inputs. While most touchscreens support some gestures, gesture-centric interfaces tend to lack visual interface elements that provide the same functionality. They are often a step backwards in usability because they have many of the
drawbacks of a command line interface: [NN10] commands can be hard to remember, impossible to explore, and hard to debug. There are no visual cues for what gestures the system will recognize, and there is no standard set of gestures for all functionality. Some work has been done to standardize gestures to make them more intuitive [Nor09], but adoption of standards continues to be low with few exceptions like those adopted into smartphone interactions. Research suggests that multi-touch interfaces have usability gains similar to command-line interfaces for expert users [For07] [Tan02] [Nor09], but unfortunately, smartphone gestures are usually one or two-fingered actions that are very simple to perform, and do not exploit multi-touch capabilities. This leaves simple one-finger target selection as by far the most common touchscreen interaction. Because of this, this dissertation will focus exclusively on target-selection errors and does not make any claims about gestural input errors.

2.1.1 Input Accuracy on touchscreens

In terms of time efficiency, touchscreen interactions are competitive with mouse-driven input [SS91]. In some contexts, such as bi-manual or multi-touch actions, they can even be significantly faster [For07]. However, touch interactions are significantly more error prone. This trade off between speed and accuracy is well known and is described formally by Fitts’ Law [Fit54] [Mac92]. A common form of this equation is as follows:

\[ MT = a + b \log_2 \left( \frac{A}{W} + 1 \right) \]

where \( MT \) is the Movement Time, \( a \) and \( b \) are empirically derived constants, \( A \) is the amplitude as measured by the distance between targets, and \( W \) is the width of the target. \(^1\)

To summarize, the speed at which a person can accurately select a target is governed by the size of the target and the distance that he or she has to move. Making targets larger or putting targets closer together improves performance. Our concern is with error, and unfortunately Fitts’ Law does not model error directly; strictly speaking, Fitts’ Law is based on a statistical assumption that the error rate in target selection will be no higher than

\(^1\)There are multiple forms of Fitts’ Law, including updates and refinements, but this equation sufficient for our discussion.
4% [Wob08]. For one-handed mobile phone use, this corresponds to about one in every twenty-five touches. However, empirical evidence shows that touching some areas of the screen can reach error rates of close to 30% [PH08] [Par06] or even above 40% [Hen11].

Touchscreen accuracy continues to be the most critical problem in mobile usability because of the frequency in which people now use their phones. A brief overview of solutions that mitigate the occurrence of errors are discussed below.

### 2.1.2 Solutions to mitigate errors in touch paradigm (Usability)

Past work on maximizing performance of touch-based interaction roughly correspond to design-time and run-time techniques for error identification and recovery. Guideline solutions (design-time) focus on developing a visual interface design that complements the interaction style. Auxiliary solutions (run-time) computationally identify potential errors and uses heuristics to recover from them.

#### 2.1.2.1 Guidelines

Guideline solutions take insights from Fitts’ law and ergonomics to present an interface with the fewest usability concerns. This can be difficult because changing environments can drastically change some design elements such as color palettes and layouts. Moreover which hand you use, one handed or two handed and how you grip the phone changes the optimal size and position of targets [Par06] [PH10] [Ng14].

The Nielsen Norman Group maintains a document that enumerates in depth usability studies and corresponding design guidelines [Gro17b]. A well-designed touch interface will never have a target that is smaller than the average finger pad, about eight millimeters in diameter, although many will be larger to accommodate more dynamic environments. Additionally, gaps between targets lessen the occurrence of “fat finger” touches. Content should fit completely on the screen or should be reduced in size. Making these changes will not just improve people's ability to perform accurately, they will improve users’ overall mobile experience.
2.1.2.2 Auxiliary Solutions

Some types of typing errors can be identified by the system but others (e.g., miskeying a date) are harder. When an error is made on a soft keyboard, users are less likely to notice that an error has occurred than when using a physical keyboard [Bre07]. Most mobile typing interfaces have a dictionary that is checked against user input. If a word is not in the dictionary, it often assumes that you have made an error and corrects it to the closest match without user intervention [All11]. This approach has nothing to do with the activation point of any one touch. Instead, errors are determined after a word has been completely typed. Letter by letter solutions have also been employed to minimize compounding errors. Some solutions will visually highlight the most probable next key [AF09] [MZ08] [Mag04] while others will dynamically resize the activation areas of keys based on the probability of its occurrence [Rud11] [Him03] [Goo02] [Wil08]. Auxiliary solutions are not commonly integrated into task interfaces, but could be useful when enough contextual information is available.

2.2 Task Modeling in HCI

Cognitive models (also referred to as cognitive engineering models) provide qualitative and quantitative evidence that can be used as a basis to find measurable improvements to human-computer interaction and user experience tasks which often lack engineering-based approaches. This interdisciplinary field integrates data and knowledge from psychology, physiology, and computer science into quantitative computing environments that predict user performance. Designers and user experience teams can use cognitive modeling to iteratively test interfaces throughout the development process before ever showing a system to human subjects. In high-stakes domains such as aviation, cognitive modeling has been an effective tool to test the limits of human capabilities without the financial cost and adverse-outcome possibilities of building prototypes [McN00] [GC02] [BK05]. Current models can capture aspects of performance such as working memory limitations, performance under sleep deprivation, learning effects, and other cognitive phenomena.

Models are comprised of primary tasks common in human-computer interaction such as reading, typing, forming goals, clicking on buttons, etc. There are many different mod-
eling frameworks, each with different strengths and weaknesses. Some models focus on representing detailed aspects of task while others focus on representing the brain and how users process information. Card, Newell, and Moran's seminal publication, *Psychology of Human-Computer Interaction* [Car83] formally introduced three engineering modeling frameworks: The Model-Human-Processor, GOMS, and the Keystroke Level Model, all of which became integral to the cognitive modeling community. A basic introduction to these frameworks and how they work is provided in order to situate further discussions.

2.2.1 Engineering Models

2.2.1.1 Model Human Processor

The Model Human Processor describes the user in terms of processors and memory with three components: the perceptual system, cognitive system, and motor system. The perceptual system represents sensory information as input from the environment with visual and audio buffers as necessary. The cognitive system has specifications of working memory and long term memory complete with some psychological phenomena like perceptual decay and the limits of working memory. The motor system interacts with the environment through hand and head movements.

This specification has become extremely influential. The delineation of the user into perceptual, motor, and cognitive pieces also underpins both ACT-R and EPIC, two highly regarded cognitive architectures (discussed further in Section 2.2.2). Additionally, GOMS, which represents tasks in terms of the actions and goals, integrated this paradigm to create CPM-GOMS (CPM stands for Cognitive-Perceptual-Motor).

2.2.1.2 GOMS

GOMS is designed to compare alternative strategies to complete a task. The acronym stands for Goals, Operators, Methods, and Selection. The goals are objectives to be accomplished using the available operators (i.e. actions) that a user can perform. Methods consist of the goals and subgoals necessary to complete the task. When a goal can be accomplished more

---

2Some consider the MHP to be complex enough to be an architecture on its own, but it does not have an established implementation popular in the research community. Since it was introduced in The Psychology of Human-Computer Interaction, it is placed with the other frameworks introduced at the same time.
than one way, the path branches with every alternative method. When the model encounters a branch, selection rule choose which path is taken based on contextual heuristics. Since its introduction, variations like NGOMSL (Natural GOMS Language) [Kie94] and CPM-GOMS [JG95] have been created to focus on a particular aspect of a task.

CPM-GOMS focuses on the interactions between the cognitive, perceptual, and motor processors and is the chosen framework for this dissertation. The acronym CPM has two meanings: Cognitive-Perceptual-Motor and Critical Path Method. Much like the MHP, CPM-GOMS breaks the task down into a very low-level representation as information availability and physiological constraints determine when operators are available at any given time. The critical path in a model is defined by the operators that determine how long the task takes to complete. For example, a user may need to read information while clicking a button simultaneously. It may take the user 200 milliseconds to perceive information on the screen but 1200 milliseconds to move the mouse to the button. In this case, the 1200 millisecond motor movement is on the critical path. Any change to this operator changes the completion time of the task.

2.2.1.3 Keystroke Level Model

The keystroke level model, sometimes referred to as KLM-GOMS, is a simplified GOMS model that cannot branch. It is a precise description of the physical actions necessary to complete a task such as clicking a mouse button or rehoming one's hands to the keyboard. Multiple methods are not considered, so selection rules are unnecessary. This leaves one main goal as the task, and operators to complete it. Of all the modeling techniques discussed in this section, it is the most straightforward and least complex approach to predict performance. However, its simplicity creates limitations. It can only predict expert performance without errors, because it cannot take into account any mental phenomena such as learning, recall, working memory limitations, or usability [Hei08].

2.2.2 Architectures

The Psychology of Human-Computer-Interaction was published in 1983 when computing power was extremely limited. As the field began to explore the theories of cognition, it became apparent that more in depth frameworks were necessary to explore more complex
mental phenomena [OO90]. As computing power grew, researchers began unifying different theories of cognition together into single systems. These systems became what we call cognitive architectures. The most popular architectures are ACT-R [And96], Epic [KM97], and Soar [Lai87] [New90]. All three architectures revolve around a system of production rules for cognitive decision making, but they have very different underlying structures. None of the cognitive architectures claim to be directly analogous to theories of biology, and different architectures are more appropriate to model certain kinds of tasks.

2.2.2.1 Soar

Newell’s Soar cognitive architecture was one of the first computational systems to be presented as a unified theory of cognition [New90]. Unlike the other architectures discussed in this section, it does not differentiate the perceptual, motor, and cognitive systems. Instead, it represents the world as a problem space in which each possible operator will change declarative memory (how state information is stored). During the decision cycle, all productions that match the current state of declarative memory will fire. Once the first round of productions have been resolved, the state has likely been changed, and more productions may match and fire in the next round. This will continue until no productions match the current state. At this point, a decision procedure fires, and the process repeats. Preference heuristics determine what the next action will be, but if there is any ambiguity as to which action should be taken, the architecture will randomly choose between the possible actions. A new production is created to handle this case, and thus Soar can “learn”.

2.2.2.2 EPIC

The EPIC cognitive architecture was one of the first to focus on the perceptual and motor components of interaction. Rather than modeling the physics of movement (torque, gravity, light, etc.), Epic describes the time necessary to complete an action. Each action is divided into a preparation phase and an execution phase. The preparation phase is the reaction time after the cognitive processor requests the action, and the execution phase is the time necessary to physically complete it. When the production cycle begins, all of the productions that match declarative memory will fire. All of the processors run in parallel to each other, and the cognitive processor is internally parallel. However, people have only one set of eyes
and hands, so the perceptual and motor processors are effectively serial. Epic has a special kind of memory called “executive knowledge” that regulates parallel tasks. This focus on simultaneous processors make it a good architecture for modeling multitasking. However there is no learning mechanism in EPIC.

### 2.2.2.3 ACT-R

ACT-R is by far the most widely used architecture currently in cognitive modeling. Its creation was motivated by approaches in cognitive neuroscience to simulate cognitive performance [And83] [And96]. Because of this focus, it has a more detailed representation of memory than any of the other architectures mentioned. In order to interact with the environment (interfaces), it had a basic serial interactive layer known as the Visual Interface [MD95]. This layer proved to be inefficient for simulating high-performance tasks that require parallel input/output streams. Eventually, the Visual Interface was replaced with a perceptual/motor layer inspired by EPIC’s. The resulting architecture (dubbed ACT-R/PM) provided a rich environment for modeling a broad range of tasks [BA97].

### 2.3 Cognitive Modeling User Interfaces

All of the modeling techniques described so far require the modeler to have a to have proficient knowledge of both psychology and computation before ever writing a model. However, interface designers, the people that would benefit the most from writing models, are not likely to have this background. In order to increase the number of people who use cognitive modeling, there has been a push in the last decade to make it more accessible to non-technical users. The most well-known of these is Cogtool, a WYSIWYG interface for ACT-R [TJ08]. This work requires us to examine and modify the interactions between the perceptual, motor, and cognitive processors, which none of the cognitive architectures are designed to do. Therefore, this section will discuss interfaces for GOMS, the framework used in this dissertation.
2.3.1 Influential GOMS User Interfaces

There have been several projects aimed at reducing the technical qualifications necessary to make from cognitive models. All of the GUIs discussed here still required the user to know how to break down their task into a KLM-level representation, but provide the user with additional information or functionality. We will focus specifically on GOMS GUIs that gained traction in the research literature but are not publicly available or no longer actively maintained.

GLEAN, developed by the same creator of the EPIC cognitive architecture, was one of the earliest front end programs designed for GOMS evaluations [Kie95] [Kie99]. The user supplied benchmarking tasks and a GOMS model written in GOMSL, a custom variation of GOMS. GLEAN in turn extracted static metrics of usability such as learning time. Since GOMS describes the ways in which a task can be completed, GLEAN allowed users to benefit from GOMS evaluation techniques without needing to compute them themselves.

Apex was specifically designed to be useful in domains such as air traffic control and developing reliable autonomous software agents [Fre99]. Its goal was to manage the cost of preparing, running, and analyzing simulations in CPM-GOMS. Unlike GLEAN, Apex actually automated part of the modeling process. It took input in the higher-level goal-based representation, and used behavioral primitives (templates) to create the low-level CPM-GOMS model [Joh02]. The resulting system did not require a great deal of specialized expertise to create accurate zero-parameter models [Mat03].

SANLab-CM, released over 10 years after Apex, was developed by Patton and Gray from Rensselaer Polytechnic Institute to be a fully visualized GUI for CPM-GOMS models [PG10]. The user viewed and manipulated pieces of the models through a drag and drop interface that clearly showed the connections and the operational order of the nodes. In addition, it added a mechanism that introduced stochastic probability to selection rules. The stochastic behavior created a range of reasonable completion times and more correctly identified critical paths in CPM-GOMS models. The user only needed to know the KLM representation of their task, find the corresponding actions in the toolbox, and drag the goals together to create the model.

Maintaining research software can be a Sisyphean task. Operating systems are constantly being updated and porting a program from one version to the next is not always
feasible or practical. In academia, personnel turns over every few years and priorities shift as funding situations change. All of the above mentioned programs have become unavailable or unusable, presumably due to a lack of continued funding over time.

### 2.3.2 Cogulator

The above projects directly influenced the design and features of Cogulator [Cor14]. Although it was developed at a research company, its creation is not currently associated with an academic publication.\(^3\)

Cogulator is a Human Performance Calculator motivated by the need for a modeling GUI for GOMS with an integrated theory of working memory [Est15]. It is currently being used to model air traffic control interfaces. Like GLEAN, the user inputs a text-based model defined in goals and subgoals, but it also integrates preformatted snippets of code like Apex's templates. As with SANLab-CM, you can view the model graphically. It creates a Gantt chart that shows clearly if your text-based model is producing the correct order of actions.

### 2.4 Human Performance Errors

The study of “human error” can be difficult because by definition, the problem lies with the person, not the environment. In practice, when something is labeled as an error, it often marks the end of the social and psychological process of causal attribution [WC03]. One way to make progress in defining error is to look at it through the lens of usability. In many domains, iterative design is not possible and empirical evidence is difficult to obtain due to the high-cost potential of an error. In particular, aviation and medicine have been able to identify where problems might occur or provide explanations for observed behavior by utilizing cognitive models without putting anyone in danger.

Before delving into different kinds of errors, it is first necessary to discuss elements of correct behavior. Rasmussen describes three levels of behavior that contribute to overall correct behavior.\(^3\)

---

3Human Factors and Ergonomic Society featured an interview with the developer, Steven Estes, about Cogulator in their bulletin in June 2014. Speaking with the developer personally, he says that there are unpublished manuscripts in the works [SK14].
user performance: skill-based, rule-based, and knowledge-based behavior [Ras83]. Anderson created a similar delineation, but called them autonomous behavior, associative behavior, and cognitive behavior [And00]. This document will use Rasmussen's terms so that the term “cognitive” does not become ambiguous. Skill-based behavior, at the lowest level, is made up of largely unconscious decisions that control smooth sensory/motor actions such as writing. In this instance, a plan is formed, and once started, trying to stop it requires a cognitive response time. Only in fine motor tasks are error signals processed in line with skill-based behavior. The next level, rule-based behavior, is governed by our previous experiences. These behaviors are shaped by cues in the environment. Feedback correction therefore is found through the analysis of current environmental inputs. That analysis may require the highest level of human behavior, knowledge-based behavior. Knowledge-based behavior is anything that requires conscious problem solving and is the only level of which we are normally aware of our decision making process. With this delineation in mind, we can discuss how errors are situated within the larger context of human performance.

### 2.4.1 Cognitive Slips

Error is a broad term that can be further delineated into two separate categories: mistakes and slips [Rea90]. Mistakes are intentional actions that result in accidental consequences, whereas slips are unintended actions that prevent the actor from completing his or her goal. Slips have been categorized into three basic taxonomies by the deficits that cause their occurrence: failure in the formation of intention, faulty activation of schemas, and faulty triggering of schemas [Nor81] [Nor82] [Nor83]. Slips can either be knowledge-based errors or rule-based errors depending on the situation. Each of these categories and related modeling work will be discussed briefly below. The following list is not meant to be an exhaustive list of all kinds of slips, but rather to highlight existing areas of error investigation.

#### 2.4.1.1 Failure in the Formation of Intention

Ignoring mistakes (which also fall into this category), there are two main ways that faulty intentions can lead to errors: mode errors and description errors.

---

4 Schema is another term for execution of goals and subgoals inherent in performing a task.
Mode errors are possible when the same action produces different results based on surrounding context or mode. Text editors such as vim and emacs are notorious for mode errors because they have both a text entry mode and a command mode. Sarter and Woods explored mode errors particularly in the field of aviation through cognitive models [SW94]. They found that incomplete or erroneous mental models of the underlying flight systems was the primary cause for mode errors. It is recommended that clear confirmation feedback needs to be shown about what mode the system is currently in. Mode errors are a known usability issue, and system designers should avoid making systems with multiple modes [Woo94].

Description errors occur when an intention lacks specific details, or when an appropriate action is taken other than the one intended. For example, pushing the wrong button on a control panel with many possible targets. The intention “push the button” was successfully completed, but the intention didn’t specify which button. In medicine, delivering a correct dose of the wrong kind of anesthesia can also be a description error. The medical community has explored description errors through cognitive modeling to improve patient safety [Gab92]. They conclude that these types of errors can be minimized by making control panels less homogeneous, thus increasing the chance that a visual cue will correct an error before it happens.

2.4.1.2 Faulty Activation of Schema

These errors occur when steps of a task are done out of order. This can happen in two ways: a schema may be inserted before it is appropriate (errors of intrusion), or a schema can be removed (error of omission).

Intrusion errors occur when a incorrect schema is activated in the course of completing a goal. This could be environmental (external stimuli causes distraction), associative (similar tasks get confused), or out of habit, sometimes known as a capture error. An example of a capture error is opening up a browser to do something productive and ending up on your email unintentionally. Freed and Remington modeled intrusion errors in the domain of air traffic control using Apex [FR98]. They found that habit takes control when decision mechanisms fail to retrieve intention or knowledge of unusual conditions.

Omission errors occur when a schema is skipped either due to anticipation or because it
occurs after the primary goal is complete (post-completion error). For example, leaving your debit card in an ATM after receiving cash. Both kinds of errors can be predicted by goal based models of behavior such as the Interactive Activation Network (IAN) model [CS00] [CS06] and the Simple Recurrent Network (SRN) model [BP04] [BP06]. These models predict that omission errors will occur when the wrong episodic memory is retrieved, which is more likely to occur with distraction [Tra11].

2.4.1.3 Faulty Triggering of Schema

The errors in this taxonomy combines correct and incorrect actions together in different ways. When these slips are discussed, it is normally in the domain of speech errors. For example, spoonerisms are the result of false triggering errors which happen when schema activate in the wrong order [Nor83]. Blends happen when the operator is thinking about one thing, saying another, and says what they are thinking instead. Modeling these kinds of events have had implications for cognitive retrieval of speech [Baa92] [Cam82], but has had little impact on usability discussions.

2.4.2 Motor Errors

The limitations of the human-motor system are integrated into modern cognitive architectures. However, they tend to be secondary to the task rather than a subject of interest. Modelers tend to focus on expert performance which includes a model of motor behavior that predicts how fast the task can be completed accurately rather than dealing with unrelated motor errors in their models. There is a dearth of fine-grained models of motor errors in the literature which this dissertation is exploring.

Rather than focusing on suboptimal performance, related research in physiology has delved into the acquisition of skilled motor-based actions. Perhaps unsurprisingly, it has been shown that practice alone is not enough. Improving motor skills takes cognitive effort [Lee94]. Accurate mental models of our bodies, environment, and tools are key, and these models are updated over time and with practice [WF01].
2.5 Error Recovery

Error recovery has been a topic of interest for high-risk and high-stress situations. Because of the nature of errors, the ability to recover from an error under sleep deprivation [PH96] or in dementia patients [Bet08] has informed policies surrounding these issues. Cognitive models of error recovery do exist, but mostly in conjunction with the models of the corresponding knowledge-based errors primarily in medicine and aviation.

Medical errors continue to be one of the most costly kinds of errors in terms of financial repercussions and loss of human life. Medical professionals are held to the impossible standard of having perpetually flawless performance [Wu00]. This is the quintessential example of blame being placed solely on human actors and ending the process of causal attribution without considering systemic or environmental issues. More recent work in improving patient safety has shifted to encompass recovery solutions as well as preventative solutions [PC08][Coh07]. In line with this work, Patel et. al introduced a cognitive framework in which to discuss how errors are detected and corrected [Pat11].

Aviation and air traffic control have some of the most complex user interfaces with intense cognitive workloads and high cost of errors. Amalberti et. al found that pilots have more incidents resulting from errors under higher cognitive workload conditions, not because they make more errors, but because they make fewer error recoveries [AW97]. This is due to cognitive resources being tied up in the task, and thus they are less likely to notice an error has been made. Zhang and Xue experimentally found the limits of human performance on three well known pilot errors and suggested cockpit design and training modifications to make error recovery easier [ZX14].

This dissertation is the first modeling work that creates a set of generalizable, domain-independent models of error recovery. As such, it can be integrated as a secondary task into other cognitive modeling projects to create a more realistic estimation of true human performance.
2.6 Strategies

Human behavior is dictated by a myriad of unconscious cost-benefit analyses that manage perceptual, motor, and cognitive resources. The heuristics that govern these analyses are known as strategies. Strategies can be easily influenced by the environment and what information is present in it [And90] [And91]. Cognition does not happen in a vacuum, and the concept that factors outside the brain can also play a part is known as embodied cognition [Wil02]. Looking at human performance through the lens of embodied cognition sheds light on how specific strategies may be chosen.

Two main theories describe the mechanism driving the cost-benefit analyses: the minimum-memory hypothesis, and the soft-constraints hypothesis. The minimum memory hypothesis states that cognition is offloaded onto the environment in order to conserve cognitive resources even at the cost of it being less time efficient [KM94] [Bal97]. This view of cognition has limitations. Most notably, it does not account for adapting behavior over time.

Gray and Fu challenged these assertions and developed the soft-constraints hypothesis [GF04]. The core of the soft-constraints hypothesis states that time (anywhere from one-third of a second to three seconds), rather than memory-usage is the resource being conserved. It still supports memory conservation due to the time necessary to encode information into short-term and working memory and probability of that information being forgotten. It also accounts for training or top-down strategies that can influence performance. This approach also maintains that while local strategies chose a time-efficient solution, it may not translate to globally efficient behavior. When the soft-constraints hypothesis was directly compared to the minimum-memory hypothesis, the soft-constraints hypothesis was a better fit to the experimental data from human subjects [Gra06]. The soft-constraints hypothesis has shown how changes to the availability of information in the cockpit has significant effects on usability [Wal07]. In medicine, it has been used to develop procedural standards that increase patient safety [Bac12].
2.6.1 Microstrategies

Strategies choose between subunits of behavior known as microstrategies, low-level processes that describe the interactive behavior between the design of the available artifacts, and the cognitive, perceptual and motor processors. There are only a finite number of ways these processes can be interleaved, and an even smaller number that are relevant to the task at hand [GBD00].

Microstrategies complete the smallest subgoal of a larger task. Tasks can be broken down piece by piece into their separate actions, much like a KLM model. At this level (e.g. “Find a target”, “Move the mouse”, “Click the button”), there may be more than one way to interleave perceptual, motor, and cognitive processors to complete each subgoal. KLM is linear and cannot take this into account, but CPM-GOMS can be used to model precisely this level of detail. Moreover, CPM-GOMS is good for creating small standalone models of microstrategies that can be combined to create a very detailed, low-level model of a task. These models are especially useful when analyzing complex information systems such as air traffic control and sonar operator tasks.

While the microstrategies that a user chooses ultimately affect their performance, they can also tell us something about how that person interacts with system. Let’s examine different microstrategies for clicking a button with the mouse. This subtask has at least three different microstrategies, a slow mouse-click, a medium mouse-click, and a fast mouse-click. All three strategies result in the same action being performed, but the operators of the cognitive, motor, and visual system vary. In the microstrategies that Gray and Boehm-Davis identified, the fast mouse-click was identical to the slow mouse-click except that it skipped a visual verification step that took 200 milliseconds. When this information is applied to making an error, we can intuit that someone using a fast-mouse click microstrategy will be less likely to notice that they have made an error.

When determining which microstrategies fit a set of data, a two-dimensional matrix can be created with one dimension being the subgoals necessary to complete the task and the other dimension being the microstrategies that could successfully complete them. The critical path through the matrix can be searched for the best explanation for observed user behavior [ZH14]. Although individuals may complete the task differently, a well-defined task should elicit consistent microstrategies that optimally balance the resources of the
cognitive, perceptual, and motor processes [SG00b].
As one of this dissertation's contributions, we wrote and released an extension to Cogulator, an open source human performance calculator specializing in CPM-GOMS.\textsuperscript{12} Its creation was motivated by the need for a highly flexible interface to model air traffic control tasks. Although it is not currently associated with any academic publications, the developer has unpublished manuscripts in the works, and The Human Factors and Ergonomic Society featured Cogulator in their bulletin in June, 2014 [SK14].

While Cogulator is helpful for writing cognitive models, it lacked a key component of CPM-GOMS: selection. Selection rules were intentionally omitted to simplify the Cogulator syntax, thereby making it more approachable. However, we felt that the language would be stronger with the ability to create branching models. We contacted the developer during this process, and he was supportive of this endeavor. Once finished, selection was integrated

\textsuperscript{1}The code as written for this dissertation can be found at https://github.com/prgoodwin/CogulatorPlusMicrostrategies
\textsuperscript{2}The Cogulator Project can be found at http://cogulator.io/
into v3.0 of Cogulator released in August 2017.

In order to fully explore possible alternative strategies, we have given Cogulator the ability to explore multiple critical paths through stochastic branching. Using selection, the modeler can specify the probability that a branch will be chosen. The model can then be run repeatedly to get a realistic range of completion times. This feature, which is inspired by SANLab-CM, was not ready for release at this time. While the feature is useful, more work needs to be put into the user experience of how it works within the Cogulator environment.

### 3.1 Why Cogulator

CPM-GOMS [Joh88] has the ability to model microstrategies without coupling it too closely with a task. Before deciding on CPM-GOMS, we first explored using ACT-R [And83] or EPIC [KM97] to study errors. EPIC was an obvious choice because it supports parallel threads for cognitive operators. However, its complexity makes it powerful but demanding to learn and use. The goal for this work is to be accessible to modelers working in any framework, and the initial knowledge required to use Epic did not support this goal. ACT-R had a similar problem of being too complex to be able to easily separate microstrategies from the rest of the model. Ultimately, we came to the same conclusion as Gray and Boehm-Davis [GBD00] that the clarity of CPM-GOMS made it the best choice for disseminating stand-alone domain-independent modules.

Cogulator is the most recent GUI designed to simplify the process of creating models. Writing CPM-GOMS models can be tedious and error prone [Bau00]. In the 1990’s, researchers made several attempts to make the process easier including QGOMS [Bea96], CATHCI [Wil93], and GLEAN [Kie95]. Later approaches capitalized on graphical user interfaces to lower the bar of necessary initial programming knowledge by automating much of the process. Two programs, Apex [Fre99] and SANLAB-CM [PG10] gained some traction because of their focus on usability. However, all of these programs have become derelict over the years due to ever-changing computer environments and a lack of resources provided in academia. Cogulator’s design was heavily inspired by the above works, and the developer spent more than a year refining its user experience. It is still being actively used and updated by the MITRE corporation. Moreover, it runs on Adobe AIR rather than natively on Mac or
3.2 Cogulator Features

3.2.1 Text-based interface for easy manipulation

Modelers write their models using Cogulator’s text-based interface. It accommodates several GOMS variations including KLM, NGOMSL, and CPM-GOMS. The IDE provides basic syntax color highlighting, the ability to add comments, error-detection, and indentation conventions. The modeler can use the operators sidebar to insert the proper syntax at the
cursor’s current position.

Cogulator’s syntax is designed to be human readable. Goals can be given names, and operators have the ability to be given additional specifications. For example, in the line “verify display change”, the operator verify is in blue and the human-readable description, “display change” is in black. Cogulator only parses descriptions for the Hear and Say operators.\textsuperscript{3} In these cases, the description specifies the words needed to perform the action.

The hint functionality makes suggestions by looking for the logical consecutive actions, but it does not force the user to make the changes. For example, if a model specifies “Click mouse” a hint asks whether you would like to include “Hands to mouse” to account for the time necessary to move your hand from the keyboard. Over time, the modeler may gain a better understanding of how operators are interrelated.

Cogulator has the ability to add operators and modify time parameters easily without having to touch the source code. Adding an operator can be done either through the interface or by editing a plain text file. Every operator has 4 pieces: name, processor, time, and description (describes the operator itself for the menu interface, not adding additional details to an instance of the operator in a model). Once added, new operators appear in the sidebar indistinguishable from the original ones. To provide even more flexibility, the time of any individual instance of an operator can be modified in line by adding the new time in milliseconds at the end of the line in parenthesis. This would turn “Hands move-cursor” into “Hands move-cursor (301 ms)”.

\subsection{Working memory implementation}

Cogulator has a detailed implementation of working memory for estimating recall ability. When working memory is automated, it adjusts the memory load whenever the Store, Recall, Look, Search, Perceptual_processor, Listen, and Think operators are used. It estimates how much information can be reliably held in memory during a task. If the modeler is managing memory manually, he or she can assign names to chunks which will allow the program to keep track of each element. This provides a mechanism to capture behaviors like rehearsing to keep things in memory longer. If a chunk is recalled after it has left working memory, the text will turn red indicating that the information is no longer available.

\textsuperscript{3}This extension also parses Goal names as possible jump locations for GoTo statements.
Figure 3.2 Figure 4 from Gray and Boehm-Davis “Milliseconds matter: An introduction to microstrategies and to their use in describing and predicting interactive behavior”. Original caption: The location of TARGET is not known until it is perceived and verified. The CPM-GOMS prediction for time from mouse down on HOME to mouse down on TARGET is 970 msec. The critical path is indicated by bold lines connecting shadowed boxes.

3.2.3 Gantt Chart

The “CPM” in CPM-GOMS has a double meaning; it stands for Cognitive-Perceptual-Motor and also Critical-Path-Method. A critical path in a GOMS model specifies which actions determine how long the task takes to complete. In Figure 3.2 taken from Gray & Boehm-Davis’s “Milliseconds Matter” paper, the critical path is subtly indicated by boxes with shadows connected by slightly bolded lines. This notation has some ambiguity due to overlapping elements. In this figure, the first “mouse Up” box has four lines connected to it; one going in from the “START mouse Dn” and three leaving it. The critical path (the second line) passes through “attend-display-change” and “perceive-display” while actually connecting to “display-change”. The path is obfuscated further since 16 of the 20 nodes have shadows.
In contrast, the Gantt chart generated by Cogulator clearly shows the critical path through the model. Like the PERT chart, the X-Axis represents time and the Cognitive, Perceptual, and Motor processors are separated on the Y-axis. In this case, the critical path is clearly indicated by a single unbroken line. If an *also* (multitasking) thread contains the critical path, it is possible that the visualization may not be connected from start to finish. Nonetheless, the path should be clearly visible. For beginner modelers, this chart is much more intuitive than the PERT chart while still being useful to expert modelers.

Unlike the PERT chart, the Cogulator Gantt chart leaves out information dependencies between operators carried out by different processors, for example that the perceptual processor should not perceive a new cursor position until the motor processor carries out the cursor movement, but there is no line indicating that these two operators are related, nor does Cogulator enforce this kind of dependency. See below for further discussion of this point.

### 3.3 Weaknesses

#### 3.3.1 Only linear models

Cogulator, as released, did not support a mechanism for selection, an essential part of CPM-GOMS modeling. Selection was intentionally left out to make the language more approachable. Our extension is specifically designed to address this weakness. We argue that adding selection improves the utility of the Cogulator language for expert modelers.
without compromising the novice's ability to write simpler models.

### 3.3.2 No operator enforcement

Cogulator does not enforce any information dependencies. Processors are serial in their own functionality but work in parallel to each other. Hearing and Speaking must not be done simultaneously, but this is the only multimodal case that Cogulator accounts for. This means that it is possible to verify a display change before the mouse has finished clicking a button. Modelers must be aware of this limitation when multitasking and should use the Gantt chart to verify that things are starting after prerequisite operators have finished. If things do not appear correctly, the actions on the main thread may need to be switched with the actions on the Also thread.

Because the operator dependencies not enforced, the modeler must have basic knowledge of how the model must be put together. Cogulator gives suggestions on how to improve a model by checking for simple logical patterns. Some modeling software like Cogtool hide the more complex mechanics from the user by reducing tasks into a KLM-like structure of concrete actions. Apex and SANLab-CM both had premade templates that specified the lower-level details for higher level actions. However, since Cogulator was specifically designed for flexibility, the modeler is responsible for combining operators appropriately. The Gantt chart is a good tool for debugging these issues.

### 3.4 Extended functionality

This extension adds two new pieces of functionality: selection and probabilistic variance. Selection is an integral piece of CPM-GOMS. It provides a mechanism to compare alternative pathways for completing a task. Probabilistic variance is necessary in order to explore a range of realistic critical paths and completion times.

#### 3.4.1 Selection

Adding the ability to create branching behavior and loop requires a state, an if-statement, and a GoTo command. To do this, we have added five new operators: `CreateState`, `SetState`, `
If, EndIf, and GoTo.

3.4.1.1 States

The syntax for the CreateState and SetState operators is as follows:

CreateState state_name value
SetState state_name value

States function as flags. The saved value of a state can be compared to any string. The CreateState operator initializes a new state. State names must be unique and neither names or values can contain spaces or other whitespace characters.

SetState is used to change the value of an existing state. This command will overwrite the saved value with the new one. If it cannot find the state_name in question, the syntax color will turn red alerting you to the error.

3.4.1.2 If Statements

The syntax for if statements is as follows:

If state_name value
    ...
EndIf

If the state_name and value pair matches a saved state, the operators between the If and EndIf will be executed. Otherwise, it will skip to the line after the EndIf. Unlike the Goal syntax, the body of the if statement is not indented further.

3.4.1.3 GoTo Statements

The syntax for the GoTo statement is as follows:

GoTo Goal: goal_name
The goal_name must match exactly to how it appears in the model. This statement finds the first occurrence of the goal_name and continues the model from that point. If the goal is further down in the code, it will skip everything in between. If the goal appears earlier in the model it will create a loop. In order to avoid infinite loops, jumps are limited to 25 times in a single model. If that number is exceeded, the line will turn red (as if there is an error in it) and all additional GoTo statements will not be processed.

3.4.1.4 Feature Weaknesses

Without stochastic execution, models will only have a linear execution path. Some might argue that selection is not necessary because it adds an a new layer of complexity even though the model will not change. This argument ignores the labor cost of writing a model. Comparing completion paths becomes much simpler if you only have to write one model that branches with the change of a state rather than having to write three separate models.
Figure 3.5 New interface elements were added to report aggregate completion time data. The modeler enters the number of times the model should be run in a textbox and presses the associated button. Labels with the average, minimum, and maximum time appear to the right of the button.

### 3.4.2 Stochastic Variation

To add in a stochastic point, use the following syntax:

```
SetState state_name value probability
```

where probability is a number between 0 and 1 inclusively.

Adding stochastic variation allows modelers to capture some human variability. A realistic range of completion times is more representative of true user data than a single completion time based on one run of the model. The modeler can add the likelihood of a branch being taken by setting the probability of a state (flag). Then the model can be run repeatedly to see how it affects the completion time. If that branch is not on a critical path (and does not become one), then the possible actions will have no effect. Otherwise, the range of completion times will adjust accordingly.
3.4.2.1 Feature weaknesses

This feature is still experimental as the user experience of using probability in Cogulator has not been fully tested. Cogulator updates the model whenever text is changed, and as of now, the generated values are not saved. This means the model changes whenever it is refreshed. The Gantt Chart shows a possible path through the model, but not every path. This makes debugging more difficult, especially combined with multitasking. Since Cogulator does not enforce prerequisites for multitasking, it is possible that one branch may not be correct, but this kind of bug would not be caught when running in aggregate. Modelers should always fully test their code to ensure that all branches are error free before using this feature. Additionally, running a model in Cogulator takes a non-trivial amount of time. Running a model thousands or millions of times is not realistic.
COGNITIVE MODELS OF MICROSTRATEGIES ASSOCIATED WITH ERROR RECOVERY

4.1 Instrumentation

Eye-tracking on mobile touchscreen devices is not a solved problem. After reviewing related literature, we rigorously tested an experimental apparatus to ensure that at least 95% of our data would be usable for modeling purposes. We tested hardware sampling frequency, target shape, algorithms, ergonomics, and the amount of noise introduced by hand occlusion during touchscreen interactions to ensure the data was consistent and reliable.
4.1.1 Design Motivations

Eye-tracking allows researchers to collect data on natural and unconscious behaviors that shed light on search strategies, problem solving, focus shifts, and other thought processes useful for cognitive modeling [Bal03] [FB06]. The price of external infrared eye-tracking hardware has fallen significantly in recent years while techniques that utilize built-in webcams and computer-vision based algorithms have improved significantly [Hol13]. Additionally, external eye-trackers have become smaller, a necessity for working with hand held or mobile devices. These factors have contributed to the popularity of eye-tracking as a research tool and it is now being used at an unprecedented rate [Bal15; Kan14; KH14].

4.1.1.1 Sources of Error in Eye Tracking Data

The accuracy of eye-tracking data has improved over the years but it still contains non-trivial amounts of error that can be introduced during calibration, instrumentation, or post-collection analysis.

Calibration usually consists of the user looking at a specified point on the screen and mapping the screen coordinates to the visual degrees recorded by the eye-tracker. The more calibration points that are used, the more accurate the mapping is likely to be. However, the calibration process does not produce perfectly accurate results and is particularly vulnerable to environmental changes. Given a perfect calibration, the most popular infrared eye-trackers (including The EyeTribe) tout being accurate only to within 0.5° - 0.8° [SR 15; Tob12; Eye]. Calibration errors cause the data to be translated in the x and y directions and/or changed in magnification.

Instrumentation errors occur when any physical movement of either the user or eye-tracker invalidates the calibration mapping. Software solutions can compensate with head tracking algorithms, but noise is inevitably introduced at this stage. When appropriate, researchers use a chin rest to hold the user’s head steady for improved performance. If too much movement is introduced, the x, y locations of the gaze points will not be reliable.

Post-collection errors occur during fixation analysis. Once gaze points have been recorded, fixations must be separated from saccades. Different fixation identification algorithms have their own advantages and disadvantages [SG00a]. The most popular approaches employ velocity-based and dispersion-based algorithms because they are not computationally
expensive. Other methods like Hidden Markov Models and Minimum Spanning Trees may be able to account for more noise, but take much longer to run. The most common error introduced in this stage is a missed fixation, although false positives also occur.

Touchscreens interactions have added interference from the user’s hands moving in front of the screen during selections. Infrared eye-trackers work by reflecting a beam of infrared light off the user’s retina. In order to avoid eyelash occlusion, they must sit below the user’s face placed at an upward angle. This means that the device must be placed at the foot of the screen and will not work in other positions. This creates a situation in which the hands may obscure the eye-tracker's line of sight. When this happens, no gaze points are recorded.

4.1.1.2 Eye-tracking for mobile platforms

Most commercially released “eye-tracking” is actually face-tracking. Samsung marketed the Galaxy S4 as the first consumer device to integrate eye-tracking capabilities into their interface. However, it used the webcam and a basic face-tracking algorithm to see if the user’s face was in front of the screen \[Ble13\]. It did not do any analysis of pupil position.

There are two approaches to true eye-tracking in a mobile environment: camera-based, and infrared.

4.1.1.2.1 Webcam

Eye-tracking with a webcam uses computer vision to extract facial features from an image. The eyes must be identified, and the facial features must be tracked for changes in position and angle \[KEM13\]. Front facing webcams are already widely available on unmodified tablets and phones. However, computer vision is extremely computationally intensive even under ideal circumstances. For hand held devices, the image quality is substandard, the device’s processing power is limited, and mobile environments introduce ever changing variables such as lighting, motion, etc \[SK10\] \[Hol13\].

Camera-based solutions that run on mobile devices are rare in the academic literature. At least one software solution has been commercialized rather than remaining open to the research community. Umoove, a start-up in mobile eye-tracking, has patented several eye-tracking algorithms, but does not make the source publicly available \[Kem15\] \[KEM13\]. As
such, their accuracy and performance cannot be evaluated independently. Two systems for eye-tracking on unmodified tablets exist in the research literature: EyeTab from University of Cambridge [WB14], and an unnamed project from Texas University [HK12]. Both projects mount the tablet to a stationary holder to collect data. EyeTab’s algorithm achieves accuracy within an average of 7° (larger than half a tablet screen) at 12 frames per second, while the other has better accuracy at 4.42° but takes more than a second to analyze each frame. Neither can identify Points-of-Interest (POI’s) in a realistic interface task.

4.1.1.2.2 Infrared

External infrared eye-trackers are much more accurate than computer vision algorithms working on webcam images, but they have been too large to use in the mobile sphere until the release of the EyeTribe in December 2013. At its launch, it was the smallest eye-tracker commercially available and was made to sit below the screen of a Windows tablet. The company showcased a proof of concept for an android device at CES in 2014, and in January of 2015, the Android SDK was released. In a non-touch environment, the Eyetribe claims accuracy within 0.5° visual degrees.

4.1.2 Hardware Apparatus

The final experimental apparatus can be seen in Figure 4.1. We chose to use an infrared eye-tracker because it is significantly more accurate than camera-based solutions. In order to minimize instrumentation errors, we followed the example of Wood & Bulling and Holland & Komogortsev and mounted our tablet to a tripod to stabilize the system. During experiments, we asked participants to hold the sides of the tablet in order to simulate a plausible hand position for holding the device. The complete apparatus was then placed on a desk, and the participants stood during the experiment. When a mouse was required, a wireless Bluetooth mouse was provided on to the right of the tablet.

4.1.2.1 Position and ergonomics

The participants stood between 7.5 and 24 inches from the screen with the tablet approximately at chest level. They were instructed to hold their hands on the sides of the tablet
Figure 4.1 Experimental Setup: The EyeTribe eye-tracker attached to the front of the Surface Pro 2 tablet along the bottom edge using the EyeTribe’s proprietary harness and connected with a short cord in the USB on the left side. Both devices were attached to an adjustable height tripod with a universal 10” tablet tripod mount. If necessary, a wireless Bluetooth mouse was placed to the right of the tripod at a comfortable height for the standing participants.

and to select targets with their thumbs. This setup minimized the amount of interference between the users’ hands and the eye-tracker. When the mouse was required in the instrumentation study, the elevated mouse pad provided a seamless transition that did not invalidate the calibration.

4.1.3 Eye-Tracking fixation identification algorithm

The fixation-identification algorithm used for all three experiments was developed from a dispersion algorithm in the literature [Wid84] [SG00a]. It uses one visual degree as the window size and 200 ms as the minimum duration threshold.

The algorithm takes as arguments the window size for that user’s data (varies depending
on pixels per visual degree for that user), and a list of gaze points collected while the target(s) were active. If the first gaze point and the second gaze point are within a given window size of each other, both points are added to a fixation. Next it iterates through the rest of the list, keeping track of the distance between the current point and the last point, and the distance between the current point and the next point. If both distances are less than the window size, it adds the current point to the fixation. If the last distance is less than the window size, but the next distance is not, the current point is added as the last gaze point in the fixation, and a new fixation is initialized. This continues until all points except the last one have been tested. If the last point is within the same window as the point before it, it is added to the last fixation.

Next, each potential fixation is considered individually. If the total dimensions of the fixation exceed the window size, then the outermost point (the point furthest from the average point) is removed until the cluster fits within the window. Finally, if the reduced cluster lasts longer than a minimum duration threshold, then it is added to the final list of fixations.

The algorithm uses one visual degree as the window size and 200 ms as the minimum duration threshold. The pseudocode for the algorithm can be seen below.
FindFixations(G, d)

1 Fixation curFixations
2 List allFixations
3 List finalFixations
4 if G.Count < 2 // Error handling
5     return allFixations
6 if distance(G[0], G[1]) < d // first case
7     curFixations.Add(G[0])
8 for i = 1 to G.Count - 1 // standard case
9     lastDist = distance(G[i - 1], G[i])
10    nextDist = distance(G[i], G[i + 1])
11    if ! (lastDist > d and nextDist > d)
12        curFixations.Add(G[i])
13        if (lastDist ≤ d and nextDist > d)
14            allFixations.Add(curFixations)
15            curFixations = new Fixation()
16 if distance(G[last], G[last - 1]) < d // last case
17    curFixations.Add(G[last])
18 if allFixations.Count > 0 // Add last fixation if one was found
19    allFixations.Add(curFixations)
20 // Remove outermost point until fixation is smaller than window
21 foreach Fixation f in allFixations
22     f.reduceSize(d);
23 // If duration is longer than threshold, fixation is moved to final fixations
24 foreach Fixation f in allFixations
25     if f.Duration > 200
26         finalFixations.Add(f)
27 return finalFixations
4.1.4 **Instrumentation Study**

We rigorously tested our experimental apparatus to ensure that we could collect reliable, high-quality eye tracking data during touchscreen interactions for modeling purposes. We analyzed the apparatus, hardware sampling frequency, target shape, algorithms, ergonomics, and the amount of noise introduced by hand occlusion during touchscreen interactions. We then created an ACT-R model based on the data collected.

4.1.4.1 **Methods**

4.1.4.1.1 **Participants**

Twelve adults between the ages of 20 and 32 with normal or corrected to normal vision participated in this study. One person wore glasses and one person wore contacts. Participants were volunteers and were not compensated for their time.

4.1.4.1.2 **Task**

This experiment had two parts: touch input, and mouse input. During the touch portion of the task, participants were told to select targets with their thumbs leaving their hands on the side of the tablet. The cursor would not be visible during this time. While using the mouse, the cursor would be replaced with a red circular area cursor to more closely resemble the accuracy of a finger pad. Any overlap between the target and the area of the cursor would activate the target.

The user saw circle-shaped and cross-hair shaped targets that were functionally equivalent for each input type (seen in Figure 4.2). Because we were testing the accuracy of gaze data against a true location, the instructions stressed that when a target appeared, they needed to look at the target when selecting it with the given input.

The touch-input section and the mouse-input section were switched for some users to account for order effects. However, circle targets always appeared before crosshair to minimize confusion.
4.1.4.1.3 Calibration & Ergonomics

Participants assumed the position described above: standing with the tablet chest high, at arms length, with his or her face centered in eye-tracker sensors. We used a 16 point grid and the EyeTribe's UI sample calibration software to complete the calibration before context switching to the experimental window.

4.1.4.1.4 Software

The software for the instrumentation study was developed in C#'s WPF using Visual Studio 2013.¹ The EyeTribe API contains several WPF samples and works well in this development environment.

4.1.4.1.5 Measurements

To compute visual degrees, each participant’s eye level, the distance he or she was standing from the device, and the height of the device were recorded. Each gaze point generated by the EyeTribe was timestamped and recorded along with all system and user generated events in an external log file that was used to identify fixations.

¹Code for the instrumentation study is available for download at https://github.com/prgoodwin/Modeling-Mobile-Interactions-with-Eye-Tracking-Data
4.1.4.2 Results

4.1.4.2.1 Target Shape

An independent samples t-test assuming unequal variance was conducted to test if target shape affected the accuracy of the location of a fixation. There was no statistically significant results between the accuracy of fixations on circle targets (m = 1.10°, sd = 1.25) versus crosshair targets (m = 1.13°, sd = 1.00); t(1087) = 1.96, p = 0.60, α = .05.

Figure 4.3 shows the observed distances between fixation center and target center by target shape. The median values were not significantly different between circle targets (1.10°) and crosshair targets (1.13°). The graph was cut off at 4° so that a visual comparison could be made more easily. A nominal amount of data exists past 4° and can be seen in Figure 4.5.

One possible threat to internal validity is that both target shapes had clearly defined boundaries. The accidental result of this was that the circle and cross hairs targets were the same size. In hindsight, the cross hair lines should have extended to the top, bottom, and sides of the screen in order measure the effects of different categories of targets.

4.1.4.2.2 Fixation Identification

Our dispersion algorithm successfully identified fixations for 94.75% (1137) of the 1200 target screens. Figure 4.4 shows a sample screen overlaid with all gaze data collected for that target. The large circle is the shape and location of a randomized target in the experiment. The smaller points represent the location of all gaze-data collected while the target was on the screen, while the darker subset of points indicate the fixation identified by our algorithm.

Of the remaining 63 screens where no fixation could be found 60 occurred during the touch portion of the experiment. This indicates a clear relationship between touch-based input and noise that prevents fixations from being identified.

4.1.4.2.3 Fixation Accuracy and Precision

Fixations were on average 1.12° away from the target’s true location, and 95% of fixations were within 2.75° (standard deviation of 1.13). A percentage graph can be seen in Figure 4.5.
Figure 4.3 A boxplot with the means identified by the line in the center of the box, and the first and third quartile as the edges. The whiskers are placed at the 5% and 95% markers. Each data point was created by calculating the distance between the center point of the fixation and the center point of the target’s true location in visual degrees.

with all results. Distances between the center of the recorded fixation and the center of the targets true location were organized in ascending order. The percentage of the data that falls within that particular distance can be used as a confidence interval for identifying the correct point of interest.

<table>
<thead>
<tr>
<th></th>
<th>Horizontal</th>
<th>Vertical</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>0.537°</td>
<td>0.763°</td>
<td>1.12°</td>
</tr>
<tr>
<td>Precision</td>
<td>0.094°</td>
<td>0.076°</td>
<td>0.127°</td>
</tr>
</tbody>
</table>

Table 4.1 Accuracy and Precision of fixations presented in visual degrees. Fixations were found for 1137 of 1200 targets with a total of 24,673 points of gaze data. The screen was on average 12.24° wide and 6.12° high, but varied with how far away the user was stood from the tablet.
Accuracy was calculated by taking the horizontal and vertical distances between every fixation point and the center of the true target location and averaging them over all targets for each participant separately. Each participant’s results were then averaged together.

Precision was calculated by finding the Root Mean Square (RMS) of the distance between points in each fixation cluster. Each participant’s results were calculated separately, and then the results were averaged together.

### 4.1.4.2.4 Sampling Frequency

During our experiment, the EyeTribe was set to sample at 30Hz, or roughly 33ms. Of the 81,379 samples, only 32.80% are at or below the predicted time interval. However, if you allow for 5ms error, 94.45% of samples fall within range. For our purposes, 97.31% of samples were under 50ms (the minimum length of fixation) and would work for analyses. The other results can be seen in the table below.
Figure 4.5 Confidence intervals can be associated with the identified eye-tracking data based on the percentage of the data that is within a certain radius from true target location. 50% of all fixations were within 1.12° or less, and 95% of fixations are within 2.75° or less.

<table>
<thead>
<tr>
<th>Inter-sample time range (ms)</th>
<th>N</th>
<th>frequency (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-33</td>
<td>28047</td>
<td>33.80</td>
</tr>
<tr>
<td>34-38</td>
<td>52724</td>
<td>61.65</td>
</tr>
<tr>
<td>41-50</td>
<td>2442</td>
<td>2.86</td>
</tr>
<tr>
<td>50-100</td>
<td>1515</td>
<td>1.77</td>
</tr>
<tr>
<td>&gt;100</td>
<td>792</td>
<td>0.93</td>
</tr>
</tbody>
</table>

4.1.4.2.5 Implications for Experimental Interfaces

Eye-tracking can be done on touchscreen interfaces with the following design guidelines. Targets should be 2.75° apart to ensure that 95% of the data accurately identifies the area of interest. Existing tablet interfaces that have been designed for touch input will have targets
approximately $0.75 - 1.25^\circ$ in size, with a border on average of $0.25^\circ$ (for a total distance of $0.5^\circ$ between targets). In cases where eye-tracking data is collected on an existing interface that does not meet these standards, confidence levels can be associated with identified targets.

Custom interfaces can improve data quality without deviating significantly from current practices. Targets are already larger on touchscreen devices to improve selection accuracy, and inactive border regions vary widely in size. To approximate current interfaces, targets can be one visual degree in size separated by a $0.75^\circ$ border of inactive space. If the border is reduced to $0.5^\circ$ so that targets are $2^\circ$ apart, the reliability of the data falls to 91.6%, and targets $1.5^\circ$ apart will have a confidence interval of 81.3%. With these guidelines, the EyeTribe is adequate for our modeling work.

4.1.4.2.6 Preliminary ACT-R Model

The data collected in this experiment was a precursor to modeling behavior in a cognitive architecture. Most of our preliminary modeling has been with the ACT-R architecture, due to its widespread use in HCI and our familiarity with it. The ACT-R architecture is not ideal for our ultimate modeling goals, but for preliminary testing purposes it served as a reasonable first step.

We began by building an ACT-R model of the mouse task, for the circular target trials.

The mouse model has four productions, two for visual processing and two for manual processing. They fire strictly in sequence. A precondition in the production `wait-for-point` waits for the appearance of the target; when a scene change is detected, `wait-for-point` issues a request to find the new visual location. At that time `attend-point` shifts visual attention to that location. Once the visual location has been found, `move-mouse-to-point` issues a command for the hand to move the cursor to that location. When the move is complete, `click-mouse-on-point` clicks the mouse button.

The ACT-R architecture is parameterized for performance on a desktop system, with keyboard and mouse. There exists an extension for multi-touch actions [GT13], but the extension has not yet been validated with human performance data. Fortunately, a direct comparison of performance between mouse and touch actions has been carried out; Forlines et al. [For07] developed Fitts’ Law models for target selection with a mouse or with
a finger, on a tabletop display. To support our modeling efforts, we modified the ACT-R architecture to use Forlines et al.’s Fitts’ Law parameters for mouse movement.

With this change, the model described above predicts a task completion time of approximately 1375 ms, based on a simulation of the relevant experimental condition, with circular randomly placed targets. This is slightly higher than the experimentally observed mean of 1303 ms (median 1115 ms), with an error of 5.5%. (With the original, default Fitts’ Law parameters in place, the model predicts 875 ms, a 33% error; we do not attempt to explain the discrepancy.)

We built another model, again using Forlines et al.’s Fitts’ Law parameters but this time for touch. The touch model preserves the visual processing productions but replaces the mouse movement and mouse click productions with a single hand/finger movement production. The touch model predicts a task completion time of approximately 1175 ms, against the observed mean time of 1051 ms (median 922 ms), for an error of 11.8%. This is higher but potentially still useful for some modeling situations.

These results show a reasonable match to the models with respect to manual motor phenomena. Evaluating model performance with respect to gaze is more difficult. ACT-R does not model eye movement explicitly; it works with abstract shifts of visual attention. These shifts take a fixed duration of 185 ms in our model; i.e., they are insensitive to the varying distances that the participants’ eyes need to move to fixate on different targets. Further, ACT-R’s shift of visual attention is not accompanied by any external phenomenon, as would be the case for an architecture that models eye movement; we can interpret ACT-R’s 185 ms as a lower bound on time to a fixation. This means that ACT-R cannot make precise predictions of the location and duration of fixations; the models will at best be consistent or inconsistent with the experimental data.

In the mouse model, the production for moving the mouse fires at 150 ms; the vision module acquires the target at 185 ms. Although ACT-R is a discrete event simulation (i.e., it does not account for continuous mouse movement), it is reasonable to take this as implying that mouse movement may begin up to 35 ms before a fixation occurs. This is consistent with the relationship between mouse movement and fixations in the participant data; on

---

2[Sal01] presents the ACT-R extension, EMMA, that models eye movements directly. We did not use it here because we were still in an exploratory modeling phase, and this would have involved working at a more detailed level than seemed reasonable at the time.
average mouse movement begins 66 ms before fixations are identified. From the time the
target appears to the time of a fixation is 358 ms on average in the experimental data; this is
consistent with the 185 ms lower bound of both the mouse and the touch models.

These results were a promising first step that laid the groundwork for creating models
of error recovery.

4.2 User Studies

Target selection is the basis for all non-gesture touchscreen interactions. Therefore, ran a
series of user studies that isolated microstrategies involved in all stages of target selection.

Our first study was an exploratory pilot study that collected preliminary information
on error recovery strategies. We wanted to test if participants could accurately determine
the amount of errors they encountered. If there was a significant discrepancy between
what they thought and the actual value, we wanted to know if strategies were more closely
correlated with one than the other. This study also produced the information necessary to
begin modeling work.

Our second study focused on the strategies employed while searching for items on a
screen and how those strategies change under different constraints. The results of this study
gave us the information needed to create hypotheses about how users will behave in the
future and how to shorten the amount of time necessary to recover from an error.

The final study focused how visual feedback changes the microstrategies associated
with error recovery. We separated users into two categories based on the microstrategies
that they employed during a baseline performance test and analyzed how presenting visual
feedback impacted their ability to recover from errors. Using this information, we created a
set of models that describe the microstrategies associated with target selection and error
recovery.

4.2.1 Pilot study: faux errors

This exploratory study examined how users perceive error rates and whether it affected
the microstrategies that they employed. Users performed several iterations of a simple
target selection task while the perceived error rate was modified by ignoring certain touch
events. Since we were looking at largely unconscious behavior, we needed to know if users were consciously aware of factors in the system that may have affected their behavior. To this end, we asked participants to estimate the overall error rate of every round. We treated the modified error rate as an independent variable and used perceived error rate and microstrategies as dependent variables.

We expected that perceived error rate would be correlated with modified error rate, but we wanted to test whether microstrategy choice was more closely correlated to the conscious perceived error rate or to the true modified error rate in cases where there was a significant discrepancy between the two. Surprisingly, the results of this experiment show that the two variables are not closely correlated, and microstrategy choice is not correlated to either variable. Individual preference was a stronger predictor of future behavior than either perceived or modified error rates.

4.2.1.1 Methods

4.2.1.1.1 Participants

This experiment had 12 participants between the ages of 22 and 30. Those that wore glasses removed them during this experiment and no one was wearing contacts. Participants were volunteers at an open house or were familiar with the experimenter. No compensation was offered for their participation.

The data for one participant was discarded because of an unexplained artifact when perceived error rate was 100% when the actual error rate was 0%. These data points could potentially be explained by that person misunderstanding that 100% represented the amount of error instead of the amount of successful touches. However, this misconception does not appear to carry through their entire trial. Since these points are probably not representative of someone actually thinking that the error rate is 100%, this person's data was excluded.

4.2.1.1.2 Task

Participants were told that this experiment focused on how users perceive error on a touchscreen device and the task was described in detail. The experiment began with a button in the middle of the screen to anchor their vision to a starting point. Once pressed, the
round began. Each round had five targets with a randomly assigned $x$, $y$ value that could be no closer than 150 pixels apart or approximately three visual degrees. Participants had to select each target by tapping on it, which turned it grey when they lifted their finger. The round ended when all targets had been successfully selected. A sample screen can be seen in Figure 4.6. Each participant saw a total of 105 targets over 21 rounds.

We introduced artificial errors by ignoring successful touches based on a chosen percentage. There were 6 levels of introduced error, 0%, 10%, 20%, 30%, 40%, and 50%. For example, if the system introduced 10% error, one out of ten successful touches would be ignored, simulating a missed touch. This was done so that we could compare the microstrategies found in low-error environments with those in high-error environments. To familiarize the user with the task, the first three screens always had zero added error. After that, each error level would appear three times in random order. After each round, participants entered a guess for the total modified error rate that they encountered.

### 4.2.1.1.3 Fixation Identification

The same dispersion algorithm described in Section 4.1.3 was used to identify fixations in this experiment. Unlike the instrumentation study, all fixations were used for calculations. An overlay of identified fixations and events can be seen in Figure 4.7 recreated from a processed log file.

### 4.2.1.1.4 Continuous Data Monitoring

The first iteration of this study had the setup exactly as described in Section 4.1. However, we found that in a more realistic task, users focused less on their posture and ergonomic position and more on their goal. As a result, their hands would unintentionally obscure the eye tracker, causing a large gap in the data. Real time monitoring of the data stream was added to ensure that gaze data was being continuously collected. If the system lost data for two consecutive seconds, participants were not able to enter their perceived error. Instead, a message appeared telling them that gaze data was lost and asked them to readjust their position. New target positions were chosen and the round was repeated. If data collection was still an issue, the same message appeared a second time. After that, if the data could still
Figure 4.6 Sample round that participants completed for the pilot study. When successfully touched, the red target would turn grey. Once all targets had been successfully touched, the round ended.

not be collected, the calibration screen was brought to focus, and the user had to recalibrate before continuing.

4.2.1.1.5 Context-Free Grammars

Context-free grammar(CFGs) provided a mechanism to examine the order of recorded actions. Comparing user behavior was difficult because the logs did not lend themselves to be easily summarized. CFGs reduced the complexity of the data for analysis purposes. To create a CFG, perceptual events and motor events were reduced to a single letter as follows: targets-appear (t), fixation-start (f), fixation-end (e), missed-touch (m), accepted-touch (a), introduced-error (i). Other system responses can be inferred from these actions. This process turned log files into strings between fifteen and sixty characters. This representation let us compare the orderings of actions easily.
Reoccurring patterns were grouped into preliminary microstrategies. The most common identified strategies are as follows:\(^3\)

1. Fixation-start, fixation-end, fixation-start, fixation-end, etc...

2. Fixation-start, accepted-touch, fixation-end

3. Fixation-start, fixation-end, touch-event

4. Fixation-start, ignored-touch, ... accepted-touch, fixation-end

We used a script to generate many possible actions logs, allowing us to explore the logs' composition. Expansions were weighted based on that pattern's occurrence in the original logs. The first CFG we created could theoretically generate any of the log files:

\(^3\)Touch-event can be accepted-touch, introduced-error, or missed-touch.
However, this CFG does not contain enough contextual information. We know from the logs that some expansions are more likely than others. Instead, the CFG was expanded to reflect the microstrategies that we had previously identified, and production rules were further segmented to add contextual probabilities. This resulted in the following grammar:

\[
G \Rightarrow tC \\
C \Rightarrow sBeBC | \epsilon \\
B \Rightarrow Fa | F | \epsilon \\
F \Rightarrow iF | mF | \epsilon
\]

Once we felt we understood how different microstrategies interacted, the grammars were abandoned in favor of visualizations that could communicate more dimensions of the data.

4.2.1.2 Results

4.2.1.2.1 Perceived Error Rate

Figure 4.8 shows how users perceived the error rate versus the actual modified error rate. Perceived error rate differed from the modified rate by an average of 12% (sd=10%). However, a wisdom of crowds effect can be seen in the aggregate data. The overall perceived error rate was only 3% higher than the modified error rate (sd=14%) and the median was 4% higher than the modified error rate.

A linear effect is apparent along the diagonal. The Pearson product-moment correlation coefficient between the two variables is 0.7598. The linear regression model has an $R^2$ value of 0.5774. This correlation is much lower than we had hypothesized. Additionally, there was no correlation between the microstrategies participants used and perceived error rate, or the modified error rate. These results suggest that people are not consciously or
unconsciously changing their behavior based on the amount of errors that they encounter. Nonetheless, repeating microstrategies were identified.

4.2.1.2.2 Microstrategies

A dimension of the task is any subtask that can be completed with a single microstrategy. We divided our task into three distinct dimensions. We found three reoccurring microstrategies in each dimension as described below.
Dimension 1. Searching for target

*Visual-Search (VS)*: multiple fixations in a row with no touch input. This microstrategy is indicative of a cognitive decision making process about what action to perform next. VS occurs in 50% of all screens, and the number of fixations in the search versus its occurrence decreases in a logarithmic pattern. It is not correlated with error rate.

*No-Visual-Search (NVS)*: Defined as the absence of VS.

*Peripheral-Focus (PF)*: a single, unmoving fixation in the center of the screen during multiple touch-events. Unlike the other microstrategies, this is likely a conscious strategy by the user.
to keep their eyes still and only use their peripheral vision. Seen in “twitch” gaming, the user is trying to minimize reaction time by eliminating eye movement. The PF microstrategy in the second dimension is a continuation of this one.

**Dimension 2. Shifting attention away from target**

*No-Visual-Feedback (NVF)*: fixation-start, fixation-end, touch-end event. Both successful and unsuccessful touches are grouped together in this strategy, because they are cognitively the same action. The user anticipates the completion of a touch action without waiting for visual feedback on its success.

*With-Visual-Feedback (WVF)*: fixation-start, successful-touch, fixation-end, or fixation-start, unsuccessful-touch, repeat-touch, ..., fixation-end. This microstrategy is by far the most common one identified. It occurs in 96% of all screens.

*Peripheral-Focus (PF)* Shift of attention is entirely cognitive, and the eyes do not move. Can result in either NVF or WVF.

**Dimension 3. Choosing next action and error recovery**

*Success-With-Feedback (SWF)*: fixation-start, successful-touch, fixation-end. In this case, there is no need for error recovery, and the user will either return to the first dimension to choose the next target or the task will end.

*Delayed-Error-Recovery (DER)* fixation-start, unsuccessful-touch, fixation-end, ..., touch on different target. This microstrategy is defined by some indication that the user has noticed the error but has chosen to move on and come back to it later.

*Immediate-Error-Recovery (IER)* In this microstrategy, fixations can be in many places. Therefore, it is only defined by an unsuccessful-touch followed by another touch on the same target indicating that they saw the error and attempt to fix it immediately. This strategy is used twice as often as DER, and is most seen in high-error environments and accounts for 66% of all error recoveries.
4.2.1.2.3 Threats to validity

*Threat to internal validity:* The instructions stressed that the participant needed to perform this task as quickly and as accurately as possible. In practice, we found that without more constraints, participants only went as fast as they could without sacrificing accuracy. We hypothesize that briefing participants as to the purpose of the study made them more focused on what they were doing. Additionally, the interface was designed to maximize usability (large targets, far apart from each other). All errors observed were those that were artificially introduced into the task.

4.2.1.2.4 Preliminary GOMS Modeling

For this experiment, we wanted to move away from ACT-R and towards our final goal of CPM-GOMS. Following the work of Gray & Boehm-Davis [GBD00], we judged that ACT-R was too complex for our purposes; it requires commitment to many cognitive details that turn out to be less relevant to error behavior.

The original Cogulator used for this modeling had 27 built-in operators (and supported the definition of new operators), divided into the following categories:

- goals
- thinking
- seeing
- hearing
- saying
- moving

Each operator has an associated processor (cognitive, motor, or perceptual) and a duration. All durations can be modified if necessary, and custom operators can be added. We modeled seven low-error trials using only the built in operators.

To model this task, each dimension of the task has a small number of actions necessary to complete it, and those actions were broken down by processor. The only dimension
that required the motor processor was the second dimension: *Shifting attention away from target* (named for the action that differs). The cognitive and perceptual operators were used in every dimension. The resulting actions can be separated into three distinct categories: cognitive pauses, eye movements, and hand movements. Any shift of attention had a 50ms “attend” cognitive pause, and eye movements and hand movements had an additional “initiate” cognitive pause of 50ms. The default time for a saccade is 30ms. We assumed that users did not begin moving their hands before they looked at the target, but that once fixated, hand movements were done concurrently with whatever was happening in the perceptual processor. Finally, we specified the length of identified fixations.

When the length of fixations were specified, Cogulator overestimated performance time by approximately 10%. Unlike SanLab [PG10], Cogulator did not initially have any mechanism for comparing different methods based on probability. Modeling error-recovery with this data was not practical because of large variations in behavior due to the task not being constrained enough to produce consistent data. The point of this study was simply to observe the full range of possible responses to touch-screen errors. Nonetheless, we felt that Cogulator would be sufficient with our extension to model error recovery with naturally occurring errors and more consistent data obtained from a more constrained task.

### 4.2.2 Gnome Wars (Visual-Search Study)

This experiment was the first in a two-part study to identify the unconscious decisions participants made during target selection tasks in a touchscreen environment. Specifically, this first study focuses on behavior strategies employed while searching for items on a screen and how those strategies change under different constraints. The results of this study gave us the information needed to create hypotheses about how users will behave in the future and how to shorten the amount of time necessary to recover from an error.

We achieved three intermediary goals:

1. Tested potentially compounding factors that may influence *Visual-Search*

2. Created a cognitive model of *Visual-Search* in CPM-GOMS that approximates observed behavior
3. Tested an experimental design for the follow-up study.

4.2.2.1 Hypotheses

This experiment tested three hypotheses related to our research goals.

**H1:** The more goal targets on the screen, the more likely it is that the user will employ a *Visual-Search* strategy.

**H2:** When time constrained, the user is less likely to employ a *Visual-Search* strategy.

**H3:** When the main goal of the task changed from accuracy to speed, more organic errors are made.

This experiment had two independent variables: the percentage of goals on the screen and whether or not the level was time-constrained. The number of goal targets varied to be analogous to real world interface tasks:

- **One goal target:** one target among several. This is the most analogous to button selection in an interface.

- **Some goal targets:** distractors that require the user to be discriminatory. This is most analogous to multiple target selection such as check boxes in a list, where only some items apply.

- **All goal targets:** multiple targets on the screen and all of them are goal targets. This is most analogous to mass selection. For example, photos in an album.

The third hypothesis tested the experimental design directly. We wanted to verify that users prioritized speed over accuracy in some conditions. If we had successfully done changed their priorities, we would expect to see more errors.

This experiment also ensured that the interface choices elicited enough errors to provide sufficient data for error recovery strategies in the second study. Unlike the pilot study where we introduced “faux” errors, all errors in these experiments are naturally occurring, or “organic” errors. In this sense, Gnome Wars functioned as a pilot study.
4.2.2.2 Methods

4.2.2.2.1 Task

Gamification of experiments improves participant engagement and gives the experimenter a way to communicate rewards structures in real time to influence users’ priorities. The experiment integrated a simple target selection task into a game dubbed Gnome Wars.

The instructions stated that this experiment focused on unconscious strategies exhibited during target selection and error recovery on a touchscreen device. However, due to the priming effect we observed in the pilot study, the importance of errors was not overly stressed. Instead, the instructions focused on communicating the game mechanics in order to focus participants on the task at hand.

The game’s two assets, seen in Figure 4.10, were gnome faces with distinct expressions that differentiated a target as friend or enemy. Color pallets, shape, and sizes were all held constant to prevent this task being completed with peripheral vision. The background circle was added to give a visual cue of the active area and to normalize the shape.

Figure 4.10 The two gnome assets in the user studies. The happy gnome (left) functioned as a distractor. The angry gnome (right) functioned as the goal target that the participant had to locate. Color pallets, shape, and sizes were held constant to prevent this task being completed with peripheral vision. The background circle was added to give a visual cue of the active area and to normalize the shape.
the targets were reduced in size from the pilot study, from $1^\circ$ to $0.6^\circ$ in order to increase the number of organic errors. Users completed the level by tapping on all enemy gnomes which caused them to disappear. Once all enemies were gone, the level ended.

Whenever an enemy target was successfully selected, the user received 100 points. Whenever they missed the target or tapped a friendly target, 100 points was subtracted from their score. When timed, the user had 6 seconds to complete the level, but no points were added or subtracted based on completion time. Untimed levels came first in order to familiarize users with the task.

The order of levels was as follows:
- **Levels 0-3**: 1 enemy, 5 friends, not-timed
- **Levels 4-6**: 3 enemies, 3 friends, not-timed
- **Levels 7-9**: 5 enemies, 1 friends, not-timed

---

**Figure 4.11** Sample level of the Gnome Wars game used to study *Visual-Search*. Users had to tap all angry gnomes to make them disappear. Once all enemies were gone, the level ended.
Levels 10-12: 1 enemy, 5 friends, timed
Levels 13-15: 3 enemies, 3 friends, timed
Levels 16-18: 5 enemies, 1 friends, timed

4.2.2.2 Participants

Participants’ ages ranged from 19 and 33 and were recruited from a combined senior-level undergraduate and graduate level computer science class. Four completed the experiment wearing some form of vision correction. Participants were offered 5 points extra credit on a homework assignment and an opportunity to compete for a $10 amazon gift card that was given to the overall highest scorer.

25 out of 29 students successfully completed this experiment. The remaining four students were unable to finish the experiment due to eye-tracker calibration failures. Additionally, four participants’ data were thrown out due to eye-tracking noise that exceeded 5% of recorded data, leaving data from twenty-one participants for data analysis.

4.2.2.3 Software

Gnome Wars was developed in the Unity game engine, version 5.5.2. We started by modifying sample in the Unity Devkit released by The EyeTribe including a calibration scene that was not changed in any way.

4.2.2.4 Data collection

The instrumentation and ergonomics for this experiment are described in detail in Section 4.1 with the following minor modifications.

Preliminary fixation analysis was used to ensure that the eye-tracker collected continuous usable data. One of the hardest challenges for this work is ensuring that we have a continuous stream of eye-tracking information. In our task instructions, we stress that participants need to keep their hands from blocking the eye tracker during touchscreen

---

4source can be viewed at https://github.com/EyeTribe/tet-unity-devkit
5The final source code for this experiment can be found at https://github.com/prgoodwin/DissertationSourceCode.
interactions. However, some participants need more reminding than others. In our pilot study, we used a data gap threshold of two seconds. Any larger than that required the user to repeat the level. That solution did not translate well to this experiment because some levels only took two seconds to complete. Instead of using a raw value, we opted for using an estimation of total fixation coverage. We computed fixations after every level using average values from our instrumentation study for posture positions. The sum of all fixation durations had to be at least 70% of the time taken to complete the level. If it did not meet this threshold, the level had to be repeated. If repeating the level did not solve the issue, the interface gave the user the option redo the calibration process before continuing.

All gaze data, user events, and program responses were recorded in a log file for analysis. Gaze data was saved as a timestamped $x, y$ location, and the largest gaps in data collection were noted in line. Touch down and touch up events were logged as Correct (touches on enemy targets), Incorrect (touches on friendly targets), or Missed (touches caught by the background) along with the $x, y$ location of the target and the $x, y$ location of the activated pixel. The system responses consisted of logging metadata for all targets including the $x, y$ location, radius, and status as friend or enemy.

4.2.2.2.5 Post-collection analysis & adjustments

Fixations were identified by the algorithm described in Section 4.1.3.

The Eyetribe calibration exhibited significant but correctable bias during this experiment. We hypothesize that the sdk sample calibration program is more robust than the Unity calibration scene as this was the only major change to the calibration procedures between the pilot study and this experiment. The interface that we developed followed the guidelines put forth in Section 4.1.4.2.5. Because of this, the Point-of-Interest can be identified with 95% confidence up to 2.75°, as long as the bias is accounted for. In shorter levels where only one enemy is present (and perhaps only one fixation), the bias direction cannot be reliably identified. However, when a participant’s data is considered in aggregate, it becomes clear.

We applied a partially automated procedure to correct for the observed calibration errors. The computer translated the location of all fixations across all data from a single user using a minimum-distance heuristic and the true location of targets as recorded in the logs. The
results were then reviewed by hand to ensure that the algorithm gave reasonable results. Data from three participants got caught in local minima and were clearly not properly translated. In these cases, translations were applied manually with the average distance (heuristic metric) being communicated with each translation.

Before the calibration adjustment, fixations were an average of 3.29° away from any target on the screen which gave us a confidence interval of only 55% that the correct target had been identified. After the adjustment, four participant’s data could not be brought into the 95% confidence range and were thrown out. Of the remaining user data, fixations were on average 0.72° away from a target, and 97% of the data was within 2.75°. Figure 4.12 shows a visualization created from a log file of a level of Gnome Wars with fixations at their recorded coordinates and the translated coordinates after the minimum-distance heuristic has been applied. The minimum-distance algorithm is included the Appendix in Section A.1.
4.2.2.3  Results

4.2.2.3.1  Hypotheses

H1: The more goal targets on the screen, the more likely it is that the user will employ a \textit{Visual-Search} strategy.

The following table shows the observed frequencies of \textit{Visual-Search}. Recall that \textit{Visual-Search} is defined as fixating on multiple targets in quick succession, with at least one of them being a goal target, without engaging the motor system. The goal target is required as it indicates planning for a future action not currently underway. We found 73 occurrences of \textit{Visual-Search} across 71 of 378 levels (just under 20%). For analysis purposes, two outliers were removed, data points that exceeded three standard deviations (1925 ms) from the mean duration. Table 4.2 shows the break down of \textit{Visual-Search} frequencies by number of goal targets.

<table>
<thead>
<tr>
<th># Goals</th>
<th>VS Observed</th>
<th>Not Observed</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>114</td>
<td>126</td>
</tr>
<tr>
<td>3</td>
<td>33</td>
<td>93</td>
<td>126</td>
</tr>
<tr>
<td>6</td>
<td>28</td>
<td>98</td>
<td>126</td>
</tr>
</tbody>
</table>

\textbf{Table 4.2} Rate of occurrence of \textit{Visual-Search} separated by number of goal targets.

Users employed a Visual-Search strategy significantly less often when there was only one goal target. Fisher's exact test, similar to a chi-square test, computes significance between nominal values in a contingency table, but is more accurate for small sample sizes. The Fisher's exact test between the target-number conditions found that the observed rates in the one goal target condition differed from both three targets and six targets (1:3 \( p = 0.0005 \), 1:6 \( p = 0.0091 \)), but that the three and six target conditions did not differ from each other (3:6 \( p = 0.5566 \)). These results suggest that \textit{Visual-Search} is a binary decision rather than one that varies with additional input.
H2: When time constrained, the user is less likely to employ a Visual-Search strategy.

Time constraints did not affect the rate or time of Visual-Search. Thirty-nine Visual-Searches occurred during untimed levels as opposed to Thirty-four during timed levels. Mean duration of untimed VS fixations was 326ms versus 352ms for timed. Two-tailed t-test found no significant difference in mean ($t(151) = −1.014 \ p = 0.31$) or frequency ($p = 0.5787$, fisher’s exact test).\(^6\)

H3: When the main goal of the task changed from accuracy to speed, more organic errors are made.

Timing the levels had no significant effect on error rate. The average error rate for untimed levels was 4.3% versus 5.7%. A two-tailed t-test found no significant difference in error rates ($t(378) = −1.2382 \ p = 0.21$). This exposed a flaw in this experimental design. Error rate should have changed if users were prioritizing speed over accuracy, and this test was specifically included to ensure that this mechanic worked before the follow-up study. Therefore, the game mechanics were revised to fit the design considerations laid out in Section 4.2.2.3.4.

4.2.2.3.2 Model of Visual-Search in CPM-GOMS

We created a CPM-GOMS model in cogulator using the extended functionality including selection rules and probabilistic branching. The model had at least two fixations with a 20% chance each time of additional fixations.\(^7\) Fixations could last between 200ms and 375ms.

The Visual-Search model was run 500 times to get a range of probabilistic behavior. The results were aggregated and compared to the user data in Table 4.3. The average time of the model matched user data almost exactly differing by only 0.5%. The distribution of results can be seen in Figure 4.13. Both distributions peak between 500ms and 700ms with more than half of the dataset falling within this timeframe. The model shows a sharper cutoff in durations more than 800ms, with the maximum values being seen only on occasion in

\(^6\)See Section 4.2.2.3.3 for discussion of threats to validity for this condition
\(^7\)20% comes is informed by observed rates.
Figure 4.13 Distributions of all durations of *Visual-Search*. No statistical differences were found for any input conditions. Model was run 500 times and the data was aggregated.

...the 1700-1800 range for both the model and user data. The dip is caused by interference of probabilities. The model contains two mechanisms for determine completion time of a *Visual-Search*: fixation duration and fixation count. Duration of a fixation can be increased at three independent points at a rate of 40% probability. Additional fixations are added at a rate of 20% probability. In order to get above 775ms (where the frequency dips), at least three of these conditions must be true.

4.2.2.3 Threats to Validity

*Internal threat to validity:* It is likely that the design of this study did not succeed in changing the goal of some levels to speed from accuracy for all users. Six seconds was supposed to be short enough to shift users’ priorities while still providing an opportunity to recover from their errors. However, many participants pressed start button without pausing to read the instructions on the screen. Six seconds proved to be long enough that it rarely, if ever, cut the level short. Therefore, we present our results but make no strong claims about the affects of time-pressured environments on *Visual-Search*. To limit this threat in the...
<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>793</td>
<td>487</td>
<td>1765</td>
</tr>
<tr>
<td>Model</td>
<td>788</td>
<td>580</td>
<td>1835</td>
</tr>
<tr>
<td>Diff (ms)</td>
<td>-4</td>
<td>93</td>
<td>70</td>
</tr>
<tr>
<td>Diff (%)</td>
<td>0.5%</td>
<td>19%</td>
<td>4%</td>
</tr>
</tbody>
</table>

Table 4.3 Comparison of the results of the observed user data collected during the user study against the CPM-GOMS model of Visual-Search. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. No statistical differences were found for any input conditions. The model was run 500 times, and the results were aggregated to get a range of probabilistic behavior.

Figure 4.14 One possible run of the Visual-Search CPM-GOMS model with stochastic variability.

follow-up study, future design considerations are discussed in Section 4.2.2.3.4.

External threat to validity: The majority of our participants were international computer science graduate students in their early to mid-twenties. These individuals are likely more technologically savvy and heavier mobile device users than the general public. However, limited access to participants outside the computer science department made eliminating this threat impractical within the study’s completion time-frame. Therefore, these strategies are only generalizable to younger, expert user performance, but with the ubiquity of smartphones, the every day user is quickly becoming an expert.

4.2.2.3.4 Future design considerations

Finding a system that forces users to perform under time constraints without removing the ability to correct errors has been a difficult process. Simply telling people to complete the
task as quickly and as accurately as possible is not enough to ensure that they actually do so. In our pilot study, we verbally communicated this in the instructions. However, perhaps because they were well aware of the role of errors in the experiment, users only performed as quickly as they could without sacrificing accuracy.

In this study, we took a more direct approach and gave users a time limit as well as instructions. The time limit had to balance two opposing issues. In addition to forcing users to perform quickly, it could not be so short that it removed the ability to recover from errors. If the time limit was too long, it would not have any effect; if it was too short, users would not attempt recovery and simply wait for the level to end. Six seconds was chosen as long enough to complete a level with six goal targets and fix a single target-selection error. This approach did not pass the tests laid out by our third hypothesis.

Scoring mechanics remain the most direct way to influence user behavior. In this study, the score was only affected in as much as the user could not earn as many points if he or she ran out of time. A more direct scoring mechanic is proposed for the following study. For timed levels, 10 points will be deducted for every second elapsed, rounded to the nearest point instead of imposing a strict time limit. 100 points will still be deducted for Missed and Incorrect touches, so that errors are still sufficiently punitive. In untimed levels, no points are deducted for completion time, but errors will be extremely punitive. Under these new mechanics, the scoring system will directly reflect the experimental conditions.

4.2.3 Whack-a-Gnome (Visual feedback and recovery study)

This experiment is the second in a series to identify the unconscious decision participants made during target selection tasks in a touchscreen environment. The last experiment identified microstrategies used in the planning stages of target selection and examined potential influential factors. In this round, we separated users into two categories based on the microstrategies that they employed during a baseline performance test and then analyzed how presenting visual feedback impacted their ability to recover from errors.

By the end of this study we achieved the following goals:

1. Stand-alone CPM-GOMS models of all microstrategies observed during target selection.
2. Identified influential factors that can predict future performance.

3. A complete cognitive model of target selection and error recovery.

4.2.3.1 Hypotheses

The pilot study suggested that individual choice is a better predictor of error recovery strategies than the amount of error encountered in an interface. Based on these results, this experiment was designed to test two categories of users, which we are calling careful users and carefree users. Careful users tend to wait for visual feedback even under time pressure to ensure that their actions were successful. Carefree users are less consistent, often shifting their attention to the next action before they have completed the current one. We presented visual feedback in various locations to improve error recovery performance based on the following hypotheses:

H1: Careful users have improved recovery times when visual feedback is shown where the error occurred.

H2: Careful users have improved recovery times when visual feedback is shown at the likely intended target.

H3: Carefree users have improved recovery times when visual feedback is shown at the next logical action.

H4: Both careful and carefree users will have improved recovery times when feedback is shown at the error, and at the next logical action.

H5: When accuracy is prioritized, careful and carefree users are more likely to use microstrategies with more visual feedback.

H6: When the main goal of the task changed from accuracy to speed, more organic errors are made.
In all, this experiment had six conditions. Each was tested across three levels, with ten targets per level, for a total of thirty targets per condition. When the user successfully tapped the target, the gnome changed from angry to happy. The conditions varied where visual feedback was placed when a target selection error occurred as follows:

1. **Baseline**: No visual feedback was presented for errors. These three levels were used to separate them into careful and carefree users and establish baseline error recovery performance for each category.

2. **At Error Location**: A feedback asset was placed at the exact touch point location. Careful users were expected to have better error recovery performance, because the feedback object would communicate the error more quickly than simple visual verification.

3. **At Intended Target**: A feedback asset was placed on the closest target to the error, likely the one that the user intended to tap. Similar to the last condition, careful users were expected to have better error recovery performance, because the feedback object communicated the error more efficiently. We expected to see a slight improvement over the error location because the feedback asset would not be perfectly eclipsed by the users finger.

4. **At Next Logical Action**: A feedback asset was placed in the next logical action. Because there were no more than three Points-Of-Interest on the screen at any given time (two targets and a “done” button), it was possible to intuit what the user would do next. If the error was made on the first target, the feedback would appear on the second target; if the error occurred on the second target, it would appear on the “done” button; if it occurred on the “done” button, it would appear on the “done” button again. If feedback had to be shown on the “done” button before the other targets had been selected, the feedback asset was changed to a red circle with a line through it to indicate that it was not ready yet. Carefree users were expected to have shifted their attention to the next action, and would therefore be more likely to see the feedback placed at the next logical location.

5. **At Error and Next Logical Action**: This condition combined conditions two and four.
We hypothesized that both carefree and careful users would have improved recovery performance since multiple microstrategies were being addressed simultaneously.

6. **Accuracy Condition:** This condition did not test visual feedback. Instead, it tested how error recovery microstrategies were affected by prioritizing accuracy over speed. No visual feedback was shown during this condition.

The fifth hypothesis, taken directly from the Gnome Wars study, tested the experimental design directly to ensure that we had successfully changed users’ priorities.

### 4.2.3.2 Methods

#### 4.2.3.2.1 Task

Gamification of experiments improves participant engagement and gives the experimenter a way to communicate reward structures in real time to influence user’ priorities. This experiment integrated target selection and error recovery tasks into a game dubbed “Whack-
A-Gnome™ that tests various forms of visual feedback. Like the well known Whack-A-Mole game, users had to tap on the gnomes as soon as they popped up on the screen. Unlike Whack-A-Mole, there were only two gnomes active at any given time, and they only disappeared after being “whacked”.

The instructions stated that this experimented focused on unconscious strategies exhibited during target selection and error recovery on a touchscreen device. However, due to the priming effect we observed in the pilot study, the importance of errors was not overly stressed. Instead, the instructions focused on communicating the game mechanics in order to focus participants on the task at hand.

Each level had two gnome assets and a “done” button. The user had to tap on both gnomes individually, and then “done”. If the user made a target-selection error, the game gave visual feedback immediately (touch-down event) in different locations as enumerated in Section 4.2.3.1. The feedback asset drew the user’s attention by “dancing” (rotating back and forth) for 0.75 seconds before disappearing. Both feedback assets can be seen in Figure 4.16. If the feedback was placed on the target, or on the “done” button after both targets had been tapped, the 4.16.a was used. Otherwise, 4.16.b was used to communicate that it was not time to press the “done” button yet.

In the first five conditions, participants lost 10 points for every second that it took to complete the level, rounded to the nearest point. In addition, whenever an enemy target was successfully selected, the user received 100 points. Whenever they missed the target or tapped a friendly target, 100 points was subtracted from their score. No points were added or deducted for tapping on a target again after it had already been selected. The last three levels forced users to prioritize accuracy over time. No points were deducted for completion time, but users lost all their points if they made an error.
The order of levels was as follows:

**Levels 0-3:** No Visual Feedback  
**Levels 4-6:** At Error Location  
**Levels 7-9:** At Intended Target  
**Levels 10-12:** At Next Logical  
**Levels 13-15:** At Error and Logical  
**Levels 16-18:** Accuracy (No Feedback)

### 4.2.3.2.2 Participants

Participants’ ages ranged from 21 to 32 and were recruited from a combined senior-level undergraduate and graduate level computer science class. Eight completed the experiment wearing some form of vision correction. Participants were offered 5 points extra credit on a homework assignment and an opportunity to compete for a $10 amazon gift card that was given to the overall highest scorer.

30 out of 35 students successfully completed this experiment. The remaining five students were unable to finish the experiment due to eye-tracker calibration failures. Additionally, three participants’ data were thrown out due to eye-tracking noise that exceeded 5% of recorded data, leaving 27 participants’ data for data analysis.

### 4.2.3.2.3 Software

Whack-A-Gnome was developed in the Unity game engine, version 5.5.2. We started by modifying sample in the Unity Devkit released by The EyeTribe\(^8\) including a calibration scene that was not changed in any way.\(^9\)

\(^8\)source can be viewed at https://github.com/EyeTribe/tet-unity-devkit  
\(^9\)The final source code for this experiment can be found at https://github.com/prgoodwin/DissertationSourceCode.
4.2.3.2.4 Data collection

The instrumentation and ergonomics for this experiment are described in detail in Section 4.1. This experiment also used the same preliminary fixation analysis and recording methodologies from the Gnome Wars study described in detail in Section 4.2.2.2.4.

4.2.3.2.5 Post-collection analysis & adjustments

Fixations were identified by the algorithm described in Section 4.1.3.

During the experiment, we noticed a lag between when the user pressed start and when the level loaded. Because eye-tracking data was recorded during this time, the first set of targets in every level was ignored when determining bias direction and completion time. Error recovery information was not affected by this lag and is included for data analyses.

We then applied the partially automated procedure described in Section 4.2.2.2.5 to correct calibration errors. Before adjustment, only 32% of fixations were within 2.75° degrees of a Point-of-Interest. After adjustment, three participants’ data could not be brought within to 95% confidence and were thrown out. Of the remaining fixations, 98.7% of fixations were within acceptable ranges.

4.2.3.2.6 Microstrategies

We separated target selection into three dimensions, with one microstrategy completing each dimension of the task. The first dimension (D1), Visual-Search, has been described in detail in Section 4.2.2.3.2. Here, we focus on the second and third dimensions of target selection.

The second dimension of the task involves viewing and tapping a target, but varies when the user shifts his or her attention to the next action. These microstrategies determine how likely it is that an error will be noticed and corrected. We observed three microstrategies in this dimension.

Dimension 2 (D2): Shifting attention away from target

No-Visual-Feedback (NVF): fixation-start, fixation-end, touch-end event. The touch event
starts either during the fixation or just after. Both successful and unsuccessful touches are grouped together in this strategy, because they are cognitively the same action. The user anticipates the completion of a touch action without waiting for visual feedback on its success.

*With-Visual-Feedback (WVF)*: fixation-start, successful-touch, fixation-end, or fixation-start, unsuccessful-touch, fixation-end. The user continues to fixate on the target after the touch action is complete to verify the success of his or her action. When the user is still focused on a target after a touch event, he or she is more likely to notice errors more quickly and recover from them more easily.

*Peripheral-Focus (PF)*: Shift of attention is entirely cognitive, and the eyes do not move. Usually does not contain a verification step and mimics NVF. This microstrategy is a continuous strategy through all three dimensions. Often seen in “twitch” gaming, this microstrategy removes eye movements for faster performance.

The third dimension of the task involves error recovery if necessary. In the case where no error occurred, the user simply returns to D1 of the next action. In addition to those identified in Section 4.2.1.2.2, we observed two more error recovery microstrategies being used during this task: *Peck (PEK)* and *Muscle Memory (MM)*.
Figure 4.18 D2: *With-Visual-Feedback* microstrategy depicted in Gantt chart generated by Cogulator. Users verify the success or failure of their action before shifting their attention to the next action.

Figure 4.19 *Peripheral-Focus* microstrategy depicted in Gantt chart generated by Cogulator. This microstrategy cannot be broken down by dimension because there is no clear line when one dimension ends and the next begins when this microstrategy is used.

**Dimension 3 (D3). Choosing next action and error recovery**

*Success-With-Feedback (SWF):* fixation-start, successful-touch, fixation-end. In this case, there is no need for error recovery, and the user will either return to D1 to choose the next target or the task will end. The visualization of this microstrategy is identical to *WVF* since no recovery is required.

*Delayed-Error-Recovery (DER):* fixation-start, unsuccessful-touch, fixation-end, ..., touch on different target. This microstrategy is defined by moving onto another action before
**Figure 4.20** D3: *Delayed-Error-Recovery* depicted in Gantt chart generated by Cogulator. The user makes an error, notices it, but moves on to the next action before returning to recover from this error.

**Figure 4.21** D3: *Immediate-Error-Recovery* microstrategy depicted in Gantt chart generated by Cogulator. The user notices his or her error and recovers from it before continuing.

recovering from the error.

**Immediate-Error-Recovery (IER):** In this microstrategy, fixations can be in many places. Therefore, it is only defined by an unsuccessful-touch followed by another touch on the same target indicating that they saw the error and attempt to fix it immediately.

**Peck (PEK):** In addition to recovering from errors, pecking behavior is often seen when the responsiveness of the interface is in question. The user continuously repeats the action without waiting for visual feedback or cognitive verification.

**Muscle-Memory (MM):** This microstrategy is used in lieu of visual verification for users that most heavily conserve cognitive resources. No visual feedback is processed after actions.
Figure 4.22 D3: *Peck* microstrategy depicted in Gantt chart generated by Cogulator. The user continuously taps a button without cognitive pause.

Figure 4.23 D3: *Muscle Memory* microstrategy depicted in Gantt chart generated by Cogulator. The user does not process actions cognitively in between actions.

Instead, the lack of response at the end prompts the user to repeat the last set of actions exactly, offloading cognition onto the motor system.

*Other*: 7% of the data did not fall into one of these patterns. In the second dimension, when no fixation was captured before the touch, the focus-of-attention could not be identified. In the third dimension, some errors did not require a recovery action because the error occurred on a superfluous touch. In a few cases, a touch was considered to be an accidental palm touch when it occurred near the edge of the screen and more than 3 degrees away
from a target.

4.2.3.2.7 Categorization

Once our data had been cleaned, and the microstrategies identified, we analyzed the first three levels of every participant’s data to establish a baseline for recovery behavior. All errors were broken down into the three dimensions and the microstrategies tagged by hand. Users made an average of 5.8 errors (median=5, std=3.19) or 10.3%.\(^{10}\) Errors took an average 1203ms to recover from.

The frequency of the WVF microstrategy varied more than the other microstrategies, from 17% to 100%. Some users used all three microstrategies in nearly equal amounts, while others used WVF most of the time. This phenomena was used to delineate users into categories. Any user who used WVF at least half the time was considered to be a careful user while everyone else was a carefree user. Overall, fourteen users were classified as careful and thirteen users as carefree. The frequency of the D2 microstrategies are broken down by category in Table 4.4.

<table>
<thead>
<tr>
<th></th>
<th>WVF</th>
<th>NVF</th>
<th>PER</th>
<th>OTHER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall (%)</td>
<td>49</td>
<td>19</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>Careful (%)</td>
<td>68</td>
<td>12</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>Carefree (%)</td>
<td>28</td>
<td>27</td>
<td>38</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4.4 Breakdown of microstrategies in the second dimension of target selection observed during the baseline condition by user category and overall.

4.2.3.3 Results

Error recovery time was calculated by identifying an intended target for every missed touch, and subtracting the time between when that target was successfully activated (recovery

\(^{10}\)Additional touches occurred when a user pressed the done button before it was ready or when a target was tapped twice, resulting in superfluous actions. These are not considered to be errors, since the user tapped the intended target.
action) and the error itself. In some cases, missed touches occurred during superfluous actions, so no clear recovery action existed. When this happened, the recovery time was considered to be the difference between the time of the missed touch and the end time of the screen. The microstrategy assigned to these cases is “other”.

An experimental design flaw was noticed after all data had been collected. The conditions appeared in the same order for all participants. This opens up the possibility of learning and order effects that were not accounted for in the original experimental design. However, additional analyses suggest that the performance increases come from changing microstrategies rather than faster movement. See Section 4.2.3.3.3 for more details.

### 4.2.3.3.1 Hypotheses

H1: Careful users have improved recovery times when visual feedback is shown where the error occurred.

Careful users showed significant improvements in error recovery time when visual feedback was shown at the error location over both the baseline performance (baseline = 1280ms, at error = 885ms, $t(105) = -3.3393, p = 0.001$) and the intended target condition (at intended = 1154ms, at error = 885ms, $t(65) = -2.2963, p = 0.025$). Careful users took 395ms less than in the baseline, going from 1280ms to 885ms, a 31% improvement on average. In this condition, no shift of attention was identified, implying that the feedback appeared in the user’s field of view. This interrupts the verification step seen in the original WVF microstrategy accounting for the reduced recovery time.

H2: Careful users have improved recovery times when visual feedback is shown at the likely intended target.

Showing visual feedback at the intended target did not improve error recovery performance significantly over the baseline for careful users (baseline = 1280ms, intended = 1154ms, $t(116) = -0.837, p = 0.4$). Users fixated on the feedback, but the shift of attention away from and back to the target took nearly the same amount of time as the actions that verified success with no feedback at all. We made the assumption that both the error
Table 4.5 Average recovery times for careful users in conditions two and three which directly compared feedback being shown at the error location versus at the likely intended target. All t-tests are two-tailed assuming unequal variance. When feedback was shown at the error location, careful users showed significant improvement. This was not true for feedback shown to careful users at intended target.

<table>
<thead>
<tr>
<th>Recovery (ms)</th>
<th>Baseline</th>
<th>At Error Location</th>
<th>At Intended Target</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1280</td>
<td>885</td>
<td>1154</td>
</tr>
<tr>
<td>T-test with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>At Error Location</td>
<td>$t(105) = -3.3393$</td>
<td>$p = 0.001$</td>
<td>$t(65) = -2.2963$</td>
</tr>
<tr>
<td>At Intended Target</td>
<td>$t(116) = -0.8370$</td>
<td>$p = 0.404$</td>
<td>$t(65) = 2.2963$</td>
</tr>
</tbody>
</table>

location and the target location would be within the same field of visual attention. This assumption proved to be incorrect. Our results suggest users have a very limited field of attention that is focused exactly at the touch point. Table 4.5 compares the results from careful users performing the baseline and conditions one and two.

H3: Carefree users have improved recovery times when visual feedback is shown at the next logical action.

Showing feedback at the next logical action did not improve error recovery over baseline performance for carefree users (baseline = 1147ms, next = 1183ms, $t(90) = 0.2706, p = 0.787$). Users often shifted their focus multiple times in this condition.

While users noticed the feedback, there may have been a cognitive disconnect between where the feedback was located, and where the error was that needed to be fixed. Showing feedback away from the error is a non-standard practice, and our experiments only lasted a few minutes. There is a possibility that with training, this delivery of visual feedback could be an effective method of communication.

H4: Both careful and carefree users will have improved recovery times when feedback
is shown at the error, and at the next logical action.

Careful users did not respond well to feedback being shown feedback in multiple locations (baseline = 1280ms, multiple = 1217ms, $t(116) = 0.4597, p = 0.64$), and some shifted their attention after the feedback was displayed despite not needing to do so in an earlier condition. Similar to the cognitive disconnect that was seen when carefree users were shown feedback at the next action, this condition may have momentarily confused careful users. With training, this may be an effective method of communication.

Carefree users improved their recovery time in this condition. Recovery time fell 234ms from 1147ms to 913ms, a 20% improvement ($t(100) = -2.0839, p = 0.04$). In this condition, the feedback appears in the user’s field of attention, interrupting DER, making it more likely that they would fix their errors immediately. This lowered the frequency of DER from 49% to 35% and MM from 19% to 14%. The frequency of IER rose from 31% to 51%, accounting for the time savings. The PEK strategy was not employed in this condition at all. However, this is likely due to its rarity (1%), not due to visual feedback.

<table>
<thead>
<tr>
<th>Recovery (ms)</th>
<th>Baseline</th>
<th>At Next Logical</th>
<th>At Multiple</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-test with</td>
<td>$t(90) = 0.2706$</td>
<td>$t(64) = 2.1970$</td>
<td>$t(64) = 2.1970$</td>
</tr>
<tr>
<td>At Next Logical</td>
<td>$p = 0.787$</td>
<td>$p = 0.032$</td>
<td>$p = 0.032$</td>
</tr>
<tr>
<td>T-test with</td>
<td>$t(100) = -2.0839$</td>
<td>$t(64) = -2.0839$</td>
<td>$t(64) = -2.0839$</td>
</tr>
<tr>
<td>At Multiple</td>
<td>$p = 0.040$</td>
<td>$p = 0.032$</td>
<td>$p = 0.032$</td>
</tr>
</tbody>
</table>

Table 4.6 Average recovery times for carefree users in the baseline and conditions four and five which directly compared showing feedback at the next logical action versus the next action and at the error location. All t-tests are two-tailed assuming unequal variance. When feedback was shown at all logical locations, carefree users showed significant improvement. This was not true for feedback shown to careful users at only the next logical action.

H5: When accuracy is prioritized, both careful and carefree users are more likely to use
microstrategies with more visual feedback.

Carefree users showed a significant increase in the frequency of WVF microstrategy usage when the mechanics changed to prioritize accuracy over speed (speed = 0.275, accuracy = 0.74, $t(11) = 3.7081$, $p = 0.003$). Careful users, on the other hand, did not show a significant increase (speed = 0.67, accuracy = 0.44, $t(9) = -1.44$, $p = 0.183$).

Although the exact mechanism for what strategies are chosen is debated, the soft constraints hypothesis posits that time is one of the main resources conserved. Using WVF conserves time during error recovery but is a longer strategy to use. These results suggest that careful users subconsciously make the decision that the benefits of increasing the frequency of WVF do not outweigh its time cost.

**H6: When the main goal of the task changed from accuracy to speed, more organic errors are made.**

Accuracy-driven levels had significantly lower error rates ($t(26) = 7.5635\, p < 0.001$) than all other conditions, going from an average error rate of 10.3% to 2.7%. This provides some evidence that our new scoring mechanic successfully changed users’ priorities from speed to accuracy (see section 4.2.2.3.4 for discussion of experimental design).

### 4.2.3.3.2 Models of error recovery

Our results suggest that users respond to feedback in two different ways; either it appears field of attention they respond with a reduced error recovery time, or it is processed through their peripheral vision. CPM-GOMS only models abstract shifts of attention rather than a detailed representation of the human visual system. The modeler must decide whether something in the peripheral vision is processed.

For careful users, feedback presented at the error was within the visual field of attention, while carefree users needed feedback at the next logical action as well as at the error in order to be effective. All other conditions were functionally equivalent as they were processed through the user’s peripheral vision.
In the WVF microstrategy that we identified, users verified their actions with a 350ms pause when no visual feedback was shown. Visual feedback interrupted this verification step. If the user had to process visual feedback at another location (saccade and shift attention and back), the time necessary to process that information was almost equivalent to the verification step seen in the baseline performance.

We wrote five models that describe different ways that users complete target selection:

- error free performance
- careful user no feedback error recovery
- carefree user no feedback error recovery
- feedback processed through field of attention error recovery
- feedback processed through peripheral vision error recovery

It is not necessary to delineate feedback processed through field of attention or peripheral vision into user category, because all users employ the same microstrategies, just under different circumstances.

**Error Free Performance**

The model of error free performance consists of three target selections, varying the microstrategies used in D1 and D2. In addition to the Visual-Search model described in Section 4.2.2.3.2, the model probabilistically varied WVF and NVF, and attention shifts with peripheral focus. The last dimension, recovery, is unnecessary since no error was made.

The results of running the model 1000 times fit well with the user data as seen in Table 4.7 and Figure 4.24. The average time of the model underestimated the observed time by 144ms, or 8.2%. The model's distribution matches the user data. Both models peak around 1600ms with a tail on the upper end, but the model fails to capture large variations. This effect is likely due to the fact that the microstrategies are based on average performance.

**Careful user baseline error recovery**
Figure 4.24 Distributions of completion times assuming error-free performance for model and user data. Model was run 1000 times to get a range of realistic performance.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>1760</td>
<td>1021</td>
<td>3194</td>
</tr>
<tr>
<td>Model</td>
<td>1616</td>
<td>1130</td>
<td>2650</td>
</tr>
<tr>
<td>Diff (ms)</td>
<td>-144</td>
<td>109</td>
<td>-544</td>
</tr>
<tr>
<td>Diff (%)</td>
<td>-8.2%</td>
<td>10.6%</td>
<td>-17%</td>
</tr>
</tbody>
</table>

Table 4.7 Error-free target selection: comparison of the observed user data collected during the user study against the CPM-GOMS model. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model's performance and user's performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior.

Careful users employ the WVF strategy 70% of the time when no visual feedback is presented. While the error can occur on any touch, our model assumes that the error occurs
on the first touch. The first touch in the model consists of D1, VS or no VS, D2, WVF, NVF, or PER, and D3, IER, DER, or MM. If the user employs the WVF microstrategy, their action is verified with a 350ms cognitive pause. Otherwise, they must shift attention from the error to the next target and return. The additional two touches are identical to those in the error-free model.

The model was run 1000 times to get a range of probabilistic behavior. The average time of the model overestimated the observed time by 18ms, less than 1%. The results can be seen in Figure 4.25 and in Table 4.8. The model's distribution fits the user data peaking between 2250ms and 2750ms. However, the model shows fewer results on the high and low ends. This effect is likely due to the fact that the microstrategies are based on average performance.

**Carefree user baseline error recovery**
<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>2618</td>
<td>1481</td>
<td>4955</td>
</tr>
<tr>
<td>Model</td>
<td>2635</td>
<td>1590</td>
<td>5235</td>
</tr>
<tr>
<td>Diff (ms)</td>
<td>18</td>
<td>109</td>
<td>280</td>
</tr>
<tr>
<td>Diff (%)</td>
<td>0.7%</td>
<td>7.4%</td>
<td>5.7%</td>
</tr>
</tbody>
</table>

Table 4.8 Careful user baseline model assuming one error and no visual feedback vs. observed user data comparison. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior.

While the model for baseline performance for carefree users is similar in mean, it is not a close match to distribution of the observed user data. The model is identical to the careful user baseline model with equal probabilities for each strategy.

Careful users are a homogeneous category that use the same strategies whereas carefree users are comprised of everybody else. This category of users is not as consistent in comparison and is thus harder to predict. This work is able to reduce error recovery by addressing multiple microstrategies simultaneously. However, without a more constrained task, the shape of the distribution is likely not significant.

Feedback processed through field of attention error recovery

When visual feedback is processed through the user’s field of attention, the verification step is interrupted by perceiving the feedback, resulting in a reduced error recovery time. While the error can occur on any touch, our model assumes that the error occurs on the first touch. The first touch in the model consists of D1, VS or no VS, D2, WVF, NVF, or PER, and D3, IER, DER, or MM. The additional two touches are identical to those in the error-free model.

The model was run 1000 times to get a range of probabilistic behavior. The results were compared to careful users with feedback shown at error location, and carefree users
Figure 4.26 Carefree user baseline model distribution assuming one error and no visual feedback vs. observed user data distribution. Model was run 1000 times to get a range of probabilistic behavior.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>2886</td>
<td>1607</td>
<td>4630</td>
</tr>
<tr>
<td>Model</td>
<td>2881</td>
<td>1715</td>
<td>6055</td>
</tr>
<tr>
<td>Diff (ms)</td>
<td>-74</td>
<td>108</td>
<td>1425</td>
</tr>
<tr>
<td>Diff (%)</td>
<td>-2.6%</td>
<td>6.7%</td>
<td>30.8%</td>
</tr>
</tbody>
</table>

Table 4.9 Carefree user baseline model assuming one error and no visual feedback vs. observed user data comparison. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model’s performance and user’s performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior.

shown feedback in multiple locations. The average time of the model underestimated the
Feedback processed through peripheral vision model vs. user data distribution comparison. This included careful users with feedback shown at error location and carefree users with feedback shown at error location and next logical action. Model was run 1000 times to get a range of realistic completion times.

observed time by 82ms, or 3.1%. The results can be seen in Figure 4.27 and in Table 4.10. The model’s distribution, while a close to the user data, shows more trials at the peak between 2250ms and 2750ms and fewer on the high and low ends. Similar to the other models, this effect is likely due to the fact that the microstrategies are based on average performance.

**Feedback processed through peripheral vision error recovery**

When visual feedback is processed through the user’s peripheral vision, The user shifts his or her gaze to the location of the feedback and then shifts back to the location of the error. This delivery of visual feedback does not result in significant improvements over baseline performance. The model for this condition is similar to the field-of-attention
<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Human</strong></td>
<td>2678</td>
<td>1517</td>
<td>4514</td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>2596</td>
<td>1430</td>
<td>4160</td>
</tr>
<tr>
<td><strong>Diff (ms)</strong></td>
<td>82</td>
<td>87</td>
<td>354</td>
</tr>
<tr>
<td><strong>Diff (%)</strong></td>
<td>3.1%</td>
<td>5.7%</td>
<td>7.8%</td>
</tr>
</tbody>
</table>

Table 4.10 Feedback processed through peripheral vision model vs. user data comparison. This included careful users with feedback shown at error location and carefree users with feedback shown at error location and next logical action. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model's performance and user's performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior.

recovery model, but with two additional shifts of attention (to the feedback and return to the target), and a variable length cognitive pause to process the feedback.

The model was run 1000 times to get a range of probabilistic behavior. The results were aggregated and compared to careful users with feedback at intended target, next logical, and multiple locations, and carefree users with feedback at error location, at intended target, and at next logical. The average time of the model underestimated the observed time by 167ms, or 5.8%. The results can be seen in Figure 4.28 and in Table 4.11. The model's distribution fits closely with the user collected data, with both models peaking between 2000ms and 3000ms with a longer tail on the upper end.

4.2.3.3.3 Threats to Validity

**Internal threat to validity:**

- **Condition Ordering:** The conditions appeared in the same order for all participants. This opens up the possibility of learning and order effects that were not accounted for in the original experimental design. In order to check for this possibility, we analyzed the duration of microstrategies from the first speed-motivated condition to the last speed-based condition. We focused on microstrategies in the second dimension as these were the least likely to be affected by our experimental conditions. The second
Figure 4.28 feedback processed through peripheral vision model vs. user data distribution comparison. All conditions not explicitly listed in the field of attention model are covered here. Model was run 1000 times to get a range of realistic completion times.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human</td>
<td>2897</td>
<td>1314</td>
<td>5867</td>
</tr>
<tr>
<td>Model</td>
<td>2729</td>
<td>1530</td>
<td>6680</td>
</tr>
<tr>
<td>Diff (ms)</td>
<td>-167</td>
<td>216</td>
<td>-813</td>
</tr>
<tr>
<td>Diff (%)</td>
<td>-5.8%</td>
<td>16.4%</td>
<td>13.9%</td>
</tr>
</tbody>
</table>

Table 4.11 Feedback processed through peripheral vision model vs. user data comparison. All conditions not explicitly listed in the field of attention model are covered here. Table lists the mean completion time, the shortest completion time observed (minimum), and longest completion time observed (maximum) for both model and human performance. The mathematical error between the model's performance and user's performance has been calculated in the last two rows. The model was run 1000 times to get a range of probabilistic behavior.

dimension has three microstrategies: \( WVF \), \( NVF \), and \( PF \). \( PF \) does not have clearly
defined boundaries since no eye movements are present, so it was also not used in this evaluation. No statistically significant difference was found between the first speed-motivated condition and the last for either WVF \( t(478) = 1.179, p = 0.2389 \) or NVF \( t(227) = -0.0139, p = 0.9890 \). This indicates that the performance increases come from changing microstrategies rather than faster movement.

- **Task Constraints:** Subtle changes in task requirements can change the microstrategies employed by users. These results represent a simple target selection task with no additional interactions such as scrolling, negative consequences for double touching targets, and a “done” button that did not move. Caution must be observed when generalizing these results to tasks with more compounding factors.

  *External threat to validity:* Similar to the last study, the majority of our participants were international computer science graduate students in their early to mid-twenties. These individuals are likely more technologically savvy and heavier mobile device users than the general public. However, limited access to participants outside the computer science department made eliminating this threat impractical within the study’s completion time-frame. Therefore, these strategies are only generalizable to younger, expert user performance, but with the ubiquity of smartphones, the everyday user is quickly becoming an expert.

### 4.3 Summary

We present our instrumentation as a methodological contribution for others wanting to collect eye-tracking data during touchscreen interactions. Our instrumentation study thoroughly vetted the setup for consistency and accuracy.

In order to be able to identify the correct Point-Of-Interest 95% of the time, targets should be one visual degree in size separated by a 0.75° border of inactive space on all sides. If the border is reduced to 0.5° so that the centers of the targets are 2° apart, the reliability of the data falls to 91.6%, and targets 1.5° apart will have a confidence interval of 81.3%. With these guidelines, researchers can develop their own interfaces or establish confidence intervals for existing data sets.

We present a set of models specific to touchscreen interactions that describe all stages of target selection. We separated our task into three dimensions: 1. Searching for target,
Shifting attention away from target, and 3. Choosing next action and error recovery. The first study, Gnome Wars, isolated the first dimension, while the second study, Whack-A-Gnome, used the second dimension to predict behavior in the third.

The Gnome Wars study was the first in a two-part experiment that explored subtle changes in task structure and how they affected strategy choice. We hypothesized that Visual-Search was dependent on at least two factors, the number of goal targets on the screen and whether or not the environment was time constrained. Our results suggest that Visual-Search is a binary decision. When multiple goal targets are present, users are significantly more likely to use a Visual-Search strategy than when there is only a single goal.

We make no strong claims however on the effects of time constraints on Visual-Search. This study tried to balance time pressure without removing users’ ability to recover from errors. We examined error rate to evaluate whether or not our design was successful at changing users’ priorities. Fitts’ law tells us that we should have seen an increase in the error rate when speed was prioritized over accuracy, but this effect was not present. We changed our experiment in the follow up study to address this issue.

The second study, Whack-A-Gnome, changed where visual feedback was delivered to improve users’ error recovery performance by encouraging users to employ more time-efficient strategies. The data from the Gnome Wars study suggested that some users paid more attention to the effects of their actions than others (using a With-Visual-Feedback strategy). We used this observation to separate users into two categories, careful users and carefree users.

We hypothesized that careful users would have improved performance when feedback was shown at the error location and at the intended target, while carefree users would have improved performance when feedback was shown at the next action, and both categories of users would improve when feedback was shown in all of the above locations. Our results suggested that the improvements could only be gained under very specific circumstances.

We saw users react to visual feedback in three different ways. When feedback appeared directly in the user’s focus-of-attention, users exhibited a 20% to 30% improvement in error recovery time. For careful users, this was when feedback was presented at the error location and carefree users at all logical locations. If feedback was presented in users’ peripheral
vision, the time necessary to context switch to the feedback and back to the action resulted in no significant temporal improvements. For careful users this was when feedback was shown at the intended target, next action, and sometimes all logical locations. For carefree users, this happened when feedback was shown at the error, intended target, and sometimes at the next action. Lastly, we sometimes observed a cognitive pause, which indicated that users needed extra time to process information. We suspect that this is due to presenting feedback in a non-standard way which violated users’ expectations and caused a cognitive disconnect. We saw this when careful users were shown feedback in all logical locations and when carefree users were shown feedback at the next action.

With these results in mind, the next chapter discusses how designers can apply these results to touchscreen interfaces.
The final goal of this work is to provide a set of actionable items in the form of interface and interaction design guidelines. As mentioned in Chapter 2, much of the research about touchscreen errors focuses on preventative design. These guidelines do not address the cause of errors. Instead, we describe methods for delivering visual feedback for target selection errors that can shorten the time necessary for recovery.

5.1 Benefits of improving error recovery

Error recovery is most important when the user is multitasking in high-risk situations such as driving or operating machinery. Studies have compared users’ driving performance while using a phone to driving under the influence of alcohol [Str06]. Unfortunately, people have fewer reservations about multitasking behind the wheel, and it is now considered to be normal behavior.
Multitasking behavior can be described through a theory known as threaded cognition [ST08]. Threaded cognition states that while cognitive, perceptual, and motor systems can all run in parallel with each other, each one is serial internally. This means that a user can plan their next action (a cognitive action) while executing their current one (a motor action), but not while they are cognitively processing something else. In order to achieve multitasking behavior, the two tasks are interleaved.

How tasks are interleaved depends on what strategies are being employed. Generally speaking, tasks are interleaved at natural breaks at the end of subtasks [ZH14] [ST08]. However, task constraints can cause users to adopt strategies that break down the task further in order to maintain a minimum acceptable performance in both tasks [Bru09]. These new breaks happen at the boundaries of microstrategies based on a combination of the psychological constraints of working memory and the task constraints of performance. Generally, these new breaks occur during periods of low cognitive workload. Recovering from an error requires maintaining a cognitive problem state [Bor10] (a high cognitive workload task), making it likely that users will defer context switching until after working memory can be freed [IH07] [SB10]. This means that recovering from errors in a multitasking situation is done at the expense of the suspended activity. This can negatively impact performance and could be potentially dangerous if that task is high-risk (like driving). This work incrementally improves multitasking performance by reducing the time necessary to recover from errors.

5.2 Existing mobile guidelines

When Apple released the iPhone in 2007, they also released a slew of Human Interface Guidelines that drove their design principles. When the market shifted nearly all smartphones to be capacitive-touch, candy-bar style devices, it made many of those guidelines into industry standards. Apple traded legal blows with Google and Samsung over the industry adoption, but the end result was a refined design paradigm that worked well with the technology.

Apple [App17], Android [Goo17], and Microsoft [Mic17] all have Human Interface Guidelines up-to-date for the current release of their mobile operating systems. In addition,
the Nielsen-Norman Group has extensive research [Gro17a] available to support design guidelines for touch-screen interfaces on tablet and mobile devices [Gro17b].

This dissertation does not need to reproduce existing mobile interface design guidelines here. The resources above should be referenced for the most up-to-date information. The following guidelines are in addition to industry standards and do not come into conflict with established usability conventions.

5.3 Delivering feedback

5.3.1 Identifying user category

We categorized users based on their behavior during the second dimension of target selection that describes when users shift their attention to the next action. In order to replicate these methods, eye tracking data is required. However, the process can be reverse-engineered by observing a user's error recovery performance under various conditions.

The best and most accurate test for identifying users is to measure the recovery time of target selection errors with and without feedback at the location of the error. If the user shows significant improvements in their recovery time, they can be considered a careful user; otherwise they fall into the carefree category.

Simply looking at the recovery strategies in the third dimension of target selection is not enough to determine the user's category. No phenomena is clearly correlated with user category.

5.3.2 Guidelines

5.3.2.1 Interface guidelines

G1: Layout should have a logical order so that future actions can be predicted.

This guideline should not cause significant deviations over existing practices. Most tasks have a logical order of actions which is necessary for predicting the user's next focus-of-attention. This gives the designers the information required to be able to effectively deliver
feedback for carefree users at error location as well as at the next logical action.¹

**G2: Prevent users from completing subsequent actions without first correcting errors.**

This guideline encourages users to employ a *With-Visual-Feedback* microstrategy that makes users likely to notice and correct errors immediately. In linear tasks, interface elements can be deactivated until the previous action has been successfully completed.

This guideline may decrease the flexibility of the task interface. Care should be taken in the design to ensure that only one execution path exists in order to avoid frustrating users that may complete tasks differently.

**G3: Provide undo functionality.**

In cases where actions are reversible, an undo button should provide the ability to recover from errors even if there was a system response to the touch. In cases where the actions are irreversible, a confirmation or delay in which the action can be undone can provide a failsafe.

This guideline may increase the overall task time. When a confirmation is presented, the time added is equal to any interface animations, the time required to context switch to the confirmation, read the presented information (optional), and complete an additional interaction. In the case of a delay and undo, the time added is equal to only the delay. In our study, 95% of observed error recoveries were under 2500ms. If a delay is used, care should be taken so ensure that it does not compromise the user's view of the responsiveness of the interface.

### 5.3.2.2 Interaction guidelines

**G1: Visual feedback should be shown when target selection errors occur.**

Most mobile interfaces do not provide any indication that a missed touch has occurred even in instances when the touch misses all interface elements. If an error can be clearly and unambiguously identified, the user should be presented with visual feedback.

---

¹If for whatever reason order cannot be definitively determined, visual feedback should be given at the error location and nowhere else.
Our experiments showed a significant decrease in \textit{Delayed-Error-Recovery} strategy when visual feedback was presented. The feedback interrupted users’ next action by making them aware of the error in instances where they may not have originally noticed. This will reduce the compounding effects of an error.

\textbf{G2: Do not violate previously held expectations.}  
Consistency is imperative to ensure that no cognitive disconnects occur when feedback is presented. If the goal is to be adaptive, users should have an internal profile, and the interface should adapt to the profile rather than to the situation.

In our experimental conditions that presented feedback in non-traditional places, we sometimes observed a cognitive pause that indicated users needed extra time to process the information available to them. Violating expectations will likely cause confusion that will increase error recovery time rather than decrease it.

\textbf{G3: Avoid forcing the user to context switch.}  
All of the other guidelines can be distilled into this one. Context switching takes time away from error recovery, and should be avoided when optimal error recovery is required. The information available to the system may not be a complete representation of the world, but designers should weigh their confidence in the information against their design choices.

Our results suggest that context switching takes as long as processing an error with no visual feedback. Providing feedback still makes it more likely that users will employ an \textit{Immediate-Error-Recovery} strategy, lessening the chances of compound errors and cost. However, to see time performance gains, feedback must be placed in users’ field-of-attention.

\textbf{G4: Place visual feedback for target selection errors in the user’s field of attention.}  
If dealing with careful users, feedback should be placed at the location of the error. If dealing with carefree users, feedback should be placed both at the location of the error, and the next likely action. If unsure, defaulting to careful user behavior requires the least deviation from current norms.

In our experiments, careful users exhibited a 30\% improvement in error recovery time
when visual feedback was shown at the location of the error. Carefree users exhibited a 20% improvement when visual feedback was shown at the next logical location.

5.3.3 Discussion

The main objective for the development of the design guidelines is to provide actionable items for interface designers that follow the theoretical cognitive models. Mobile interfaces are still evolving, and these guidelines can improve error recovery.

The existing Human Interface Guidelines for mobile devices focus on user experience and usability in addition to consistency and branding. However, they fail to adequately address error recovery techniques. Given the inevitable percent of touches that result in target selection errors, this problem is ubiquitous to all operating systems in all applications.
Understanding touch interactions is increasingly important as they continue to dominate the way we interact with technology. Smartphones open up a new world of interaction possibilities, and their very nature requires special forethought to the issue of multitasking. Preventing errors through usability and design is the first step towards better user experience and safety, but error recovery is an integral piece of the puzzle that has not received much attention. Models derived from observed user behavior can shed light on the cognitive strategies and microstrategies at early stages of development making them more practical than large scale user studies. This work focuses on illuminating error recovery through cognitive modeling and presents practical, theoretical, and actionable results.
6.1 Summary

Target selection errors on touchscreens are an inherent issue due to the mismatch between the requirements of the technology and accuracy limitations of the human motor system. With this in mind, this dissertation asked two research questions:

- Q1: What microstrategies do people use when recovering from a target selection error during touchscreen interactions?

- Q2: How can we change touchscreen interface to reduce the time necessary to recover from the error?

The answers to our research questions have both theoretical and practical benefits. As such, this dissertation is organized around three distinct contributions:

- An extension to Cogulator that allows modelers to include ifs, states, and gotos. These new operators increase the utility of the language by adding the ability to write branches, loops, jumps, and conditionals, significantly expanding the types of models that Cogulator can handle. This feature was integrated into the main branch and is now available in v3.0. Additionally, our extension gives the modeler the ability to use stochastic branching to get a range of realistic completion times.

- A set of domain-independent models of target selection and error recovery strategies. Error recovery is largely ignored by the modeling community, and common errors and recovery strategies have not been fully explored. These models can be used as secondary tasks to better predict user performance on touchscreen devices.

- Guidelines for presenting visual feedback on a touchscreen device that minimize the time necessary to recover from errors. Visual feedback is not delivered effectively on mobile devices. This results in users not noticing that they have made an error, compounding the time and potential costs. Our work informs designers how to determine what kind of user they are working with and how to tailor to their needs.
6.2 Limitations

Target selection is the most common interaction in touchscreen environments, but microstrategy choice is highly dependent on the task constraints. While this work is domain-independent, it is not constraint-independent. We present a list of constraints that may affect microstrategy choice. This list is not exhaustive but is meant to guide potential confounding factors:

- **Layout:** The microstrategies that we observed were in a task environment that did not follow a grid or ordered pattern. Ordered interfaces may affect the behavior of *Visual-Search* and may lead to a different rate of *With-Visual-Feedback*.

- **Assets:** The assets that we used in our experiments were specifically designed in such a way that they could not be distinguished by the user’s peripheral vision, much like text. If the task does not require user to focus directly on a point-of-interest, it will likely change the rates of *Visual-Search*, *With-Visual-Feedback*, and *Peripheral-Focus* strategies.

- **Error cost:** The cost of an error to the user was very low since there was no lasting penalty for errors. As a result, users did not have any personal investment in the success of their actions. The consequences of the task may change the rates *With-Visual-Feedback* and *Immediate-Error-Recovery*.

- **More complex interactions:** Target selection was the only interaction required to complete these experiments. All information was present on the screen with no scrolling or gesture-based interactions required. Additionally, these tasks did not require the user to hold information in working memory. More complex task requirements may change which microstrategies are used in ways that are dependent on the task constraints.

- **System responses:** System response constraints are going to be highly individualized to the task at hand. For example, in our task, there was no penalty for double-tapping targets leading to more *Muscle-Memory* and *Peck* microstrategies than if a penalty was implemented.
6.3 Threats to validity

*Internal threat to validity:*

- **Gnome Wars:** It is likely that the design of the Gnome Wars study did not succeed in changing the goal of some levels from speed to accuracy. Therefore, we present our results but make no strong claims about the affects of time-pressured environments on *Visual Search*.

- **Whack-A-Gnome:** The conditions appeared in the same order for all participants. This opens up the possibility of learning and order effects that were not accounted for in the original experimental design. However, an analysis of the microstrategies exclusive to the second dimension showed that their duration did not change between the baseline and the final speed-motivated condition. This indicates that the performance increases come from changing microstrategies rather than faster movement.

- **Guidelines:** The guidelines laid out in Chapter 5 were written from the theoretical implications of the user studies and have merit as actionable items for designers. However, they have not been independently evaluated. Future work includes more work in this area.

- **General Task Constraints:** Subtle changes in task requirements can change the microstrategies employed by users. The results from all of our user studies represent a simple target selection task with no additional interactions such as scrolling, negative consequences for double touching targets, and a “done” button that did not move. Caution must be observed when generalizing these results to tasks with more compounding factors.

*External threat to validity:* The majority of our participants were international computer science graduate students in their early to mid-twenties. These individuals are likely more technologically savvy and heavier mobile device users than the general public. However, limited access to participants outside the computer science department made eliminating this threat impractical within the study’s completion time-frame. Therefore, these strategies are only generalizable to younger, expert user performance, but with the ubiquity of smartphones, the every day user is quickly becoming an expert.
6.4 Future Work

This work explores error recovery through cognitive modeling, an area largely ignored by designers and the modeling community. As such, there are many directions to extend this work.

Cogulator: The Cogulator program is open source, and the author welcomes community involvement to improve the software. Existing features of other modeling GUI’s are being integrated, and new features will come with continued use.

Only half of our extension was integrated into the main branch at the time of this writing. While the branching and looping behavior was adopted, we decided that the probabilistic functionality was not ready to be released. The user experience of that feature needs to be refined so that models do not change every time the user interacts with the interface.

Models: More models are necessary to describe more complex interactions that include scrolling, gestures, and memory constraints. Target selection is the most common touch-screen interaction, but it is not the only one. The microstrategies presented here are a good foundation on which to build a more complete set models that describe touchscr’een interactions.

User categories: Our user categories were binary; users were either a careful user or they were not. However, the carefree user category can likely be further delineated into additional categories given different kinds of tasks. More user studies are required to further explore this area.

Guidelines: The main objective for presenting guidelines is to provide designers with a new lens to view usability resulting from the theoretical implications of our cognitive models. However, the guidelines should be individually evaluated with their own user studies. Doing so would allow them to be further refined and to add additional guidelines as more issues are formalized.
BIBLIOGRAPHY


[Hen11] Henze, N. et al. “100,000,000 taps: analysis and improvement of touch performance in the large”. Proceedings of the 13th International Conference on


Ras83  Rasmussen, J. “Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models”. *IEEE transactions on systems, man, and cybernetics* 3 (1983), pp. 257–266.


APPENDIX
A.1 Calibration Bias Adjustment

\begin{verbatim}
FINDRECALIBRATIONFORALLSCREENS(x, y, z)
    // Compute the error for transforming by one unit in any direction
1    Array dst = new int[7]
2    dst[0] = ERRORFORSCREENS(0, 0, 0) // current error
3    dst[1] = ERRORFORSCREENS(1, 0, 0) // Right
4    dst[2] = ERRORFORSCREENS(-1, 0, 0) // Left
5    dst[3] = ERRORFORSCREENS(0, 1, 0) // Up
6    dst[4] = ERRORFORSCREENS(0, -1, 0) // Down
7    dst[5] = ERRORFORSCREENS(0, 0, 1) // Zoom in
8    dst[6] = ERRORFORSCREENS(0, 0, -1) // Zoom out
9    int minDist = dst.Min();
10   if minDist == dst[0])
11      return { x, y, z }  
12   else if minDist == dst[1]
13      ADJUSTALLSCREENS(1, 0, 0)
14      return FINDRECALIBRATIONFORALLSCREENS(x + 1, y, z)
15   else if minDist == dst[2]
16      ADJUSTALLSCREENS(-1, 0, 0)
17      return FINDRECALIBRATIONFORALLSCREENS(x - 1, y, z)
18   else if minDist == dst[3]
19      ADJUSTALLSCREENS(0, 1, 0)
20      return FINDRECALIBRATIONFORALLSCREENS(x, y + 1, z)
21   else if minDist == dst[4]
22      ADJUSTALLSCREENS(0, -1, 0)
23      return FINDRECALIBRATIONFORALLSCREENS(x, y - 1, z)
24   else if minDist == dst[5]
25      ADJUSTALLSCREENS(0, 0, 1)
26      return FINDRECALIBRATIONFORALLSCREENS(x, y, z + 1)
27   else if minDist == dst[6]
28      ADJUSTALLSCREENS(0, 0, -1)
29      return FINDRECALIBRATIONFORALLSCREENS(x, y, z - 1)
\end{verbatim}