ABSTRACT

HONG, KYUNG WHA. Improving Interface Usability through Model Transformation using Interaction Design Models. (Under the direction of Robert St. Amant and Emerson Murphy-Hill.)

Providing easy-to-use user interfaces (UIs) is a central issue in human-computer interaction. Many approaches and theories have been developed to design UIs, measure usability, and analyze human behavior. This dissertation describes a new approach focused on transforming interaction models to generate design recommendations and improve interface usability. The research draws on the areas of model-based usability evaluation and model-based user interface generation.

In this dissertation, I first conducted a study that replicates prior work using a model-based evaluation tool (i.e., CogTool) to evaluate a user interface as well as to justify design recommendations. The findings from this study are motivation of my work. I then present a model transformation approach that produces design recommendation that directly lead to design changes for improving usability. This approach is different from a traditional approach in that it automatically traverses the space of possible design alternatives and generates design recommendations as guidance for the designer to improve usability. The goal of this research is a framework to support the model transformation approach for improving the usability of a design during the iterative user interface development process. On the issue of representation in this approach, I demonstrate the how action graph formalism can be extended to incorporate multiple interaction models. Additionally, I provide an implemented system reflecting the approach with a state space search algorithm. An empirical study of the implemented system shows its effectiveness. The results of the study indicate that the model transformation approach can be useful in a design improvement task. A further exploratory study shows that the model transformation approach can also be used for analyzing design rationale for changes.

The contributions of this research are the following:

- Replication of prior study on model-based usability evaluation.
- A model transformation approach validated through empirical findings.
- Demonstration of models in action graphs as a basis for model transformation.
- Implementation of a system reflecting the model transformation approach.

Improving Interface Usability through Model Transformation
using Interaction Design Models

by
Kyung Wha Hong

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

2014

APPROVED BY:

Robert St. Amant Co-chair of Advisory Committee	Emerson Murphy-Hill Co-chair of Advisory Committee
Christopher Healey	James Lester

Benjamin Watson

DEDICATION

To my husband, Jeseung Oh.

And to my parents, Young Sik Hong and Myoung Ja Moon.

BIOGRAPHY

Kyung Wha Hong was born in Seoul, Korea. She received her Bachelor's degree in Computer Science and Engineering from Ewha Womans University, Korea in 2001. She earned her Master's degree in Computer Science from Korea Advanced Institute of Science and Technology (KAIST), Korea in 2004. After graduating, she worked at Samsung Electronics, Korea, for four and a half years as a software researcher. In 2008, she began her Ph.D. studies in Computer Science at North Carolina State University. During her Ph.D. years, she worked as a Teaching Assistant in Human Computer Interaction (HCI) classes. She also worked as a Research Assistant for HCI related projects funded by the U.S. National Security Agency and the Laboratory for Analytic Sciences. Her research areas include human computer interaction, usability, cognitive modeling, and phishing susceptibility profiles.

ACKNOWLEDGMENTS

I thoroughly thank my co-advisors, Dr. Robert St. Amant and Dr. Emerson Murphy-Hill for their guidance and encouragement. They both have been great advisors. Dr. Robert St. Amant, who has been my advisor from the beginning of my Ph.D. study, always gave thoughtful advice and insightful comments through my journey and led me to finish this dissertation. Dr. Emerson Murphy-Hill, who co-advised me from 2012, gave inspiring feedbacks to help improve my work. It would not have been possible to finish my dissertation without them. I would also like to thank my committee members, Dr. Christopher Healey, Dr. James Lester, Dr. Benjamin Watson, and Dr. Alexander Dean for their helpful comments to shape my research direction.

I am grateful to my friends in the Knowledge Discovery Lab. This includes Thomas Horton, Arpan Chakraborty, Prairie Rose Goodwin, Pat Cash, Shea McIntee, Sina Bahram, Lihua Hao, and Srinath Ravindran. Discussions and support from them were always exciting and helpful. I would also like to thank my fellow Korean friends at NCSU for their encouragement. I especially thank all members in Nehemiah group at Duraleigh Presbyterian Church for their sharing and prayers.

I give deepest thanks to my family: my parents, Young Sik Hong and Myoung Ja Moon, who pour unconditional love and prayers, my younger sister, Soo Youn Hong, who is my everlasting best friend, my younger brother, Junpyo Hong, who is also on the same Ph.D. journey, and my mother-in-law, Eungyu Lim, who always been supportive. Their doubtless trust and encouragement lead me to finish this dissertation. My endless love and thank goes to my husband, Jeseung Oh, for his sweetness, encouragement, and prayers. It would not have been possible to come this far without him. Lastly and foremost, I thank God for his love and guidance in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

1.1 Background

Providing easy-to-use user interfaces (UIs) is a central issue in human-computer interaction. Many approaches and theories have been developed to design UIs, measure usability, and analyze human behavior. This dissertation describes a new approach focused on transforming interaction models to generate design recommendations and improve interface usability. This research draws on the areas of model-based usability evaluation and model-based user interface generation.

There are two general approaches to evaluate user interfaces. The first is testing a user interface with real users. This approach is effective because usability is measured based on the users' interactive behavior with the system or a prototype of the system. However, performing live user testing is usually considered to be time consuming and expensive. The second approach is model-based: it examines the characteristics of the system's interface and the behavior of the user on the system, with reference to abstractions (i.e., models) that represent the user or the system (or both). This approach can be used to predict user performance without the need for live user testing. We can use such interaction models to evaluate interface usability (Kieras, 2002).

Model-based user interface generation tools generate user interfaces automatically from an abstract representation of relevant information about users and tasks. This approach helps interface designers by reducing the cost of creating or modifying designs. Models used in these approaches can be categorized into three types: task and domain models, abstract user interface specifications, and concrete user interface specifications (Szekely, 1996). My research extends past approaches by applying interaction modeling techniques to the problem of interface generation, particularly in generating UI recommendations (e.g., guidelines for design improvement).

In traditional human computer interaction (HCI) development, a user interface is designed and then evaluated, either with modeling techniques or with user testing. Techniques like user testing are usually applied in the later stages (iterations) of the development process, by observing how the user actually uses the interface; modeling techniques are often applied in earlier stages (Ivory & Hearst, 2001). Figure 1.1 shows the flow of an interface development process that includes model-based usability evaluation techniques. The core difference between what we call the "traditional" model-based usability evaluation approach and conventional user testing is that in the former case the decision for redesigning the interface is based on usability as measured by a model that predicts or simulates a user's interaction with the design (Kieras, 2002).

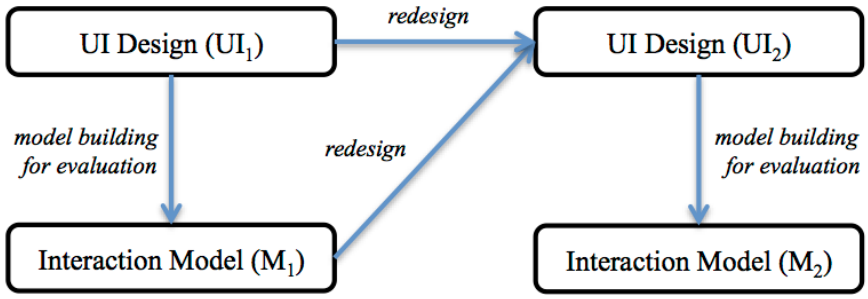


Figure 1.1 Improving design through the traditional model-based approach

In the traditional model-based usability evaluation approach, once the designer designs a UI (UI_1), the modeler or the designer constructs a model of the interface (M_1) using modeling formalisms. Having a model, the designers are able to measure some aspects of usability by the information represented by the model, such as predicted user performance (e.g., how long the model takes to execute tasks), the complexity or difficulty of interaction (e.g., whether the model can map its internal representation of the procedural task requirements to the functions provided by the UI), and learning time (e.g., how

and whether performance improves as the model gains information about how to carry out the task by interacting with the UI). From this information, the designers can make decisions about whether the UI needs to be redesigned to improve usability. If the designers makes the decision to improve the UI, the original UI design (UI_1) is modified to produce a new UI design (UI_2). Redesigning tasks can be done by redesigning the UI (UI_1), based on the designers' experience or knowledge of design principles ($UI_1 \rightarrow UI_2$). Alternatively, if the designers have a good understanding of the relationship between the design (UI_1) and the interaction model (M_1), they may be able to create a new design (UI_2) with better usability based on that knowledge ($M_1 \rightarrow UI_2$). We call the redesign steps ($UI_1 \rightarrow UI_2$ and $M_1 \rightarrow UI_2$) in this approach a *UI transformation*. However, this redesign step may take days or even weeks and may involve an arbitrarily large number of design decisions.

1.2 Motivation

One major benefit of using a model-based evaluation approach is that design iterations can be carried out at lower cost and in less time, because usability results can be obtained prior to implementing a prototype or actual user testing. Moreover, the designer can gain the insight into how the design supports the user in performing the task (Kieras, 2002). The designer can also compare usability between alternative designs in quantitative terms.

Nevertheless there are shortcomings in the traditional model-based approach to improving the usability of a user interface. Given the possible changes the designer can make in the UI Transformation stage, the result may be a UI with better usability, but this is not guaranteed; the designer may miss considering other design changes suggested by the model. Moreover, while analysis of the model M_1 provides information about the usability of UI_1, decisions about how to apply this information in order to improve usability in UI_2 are entirely up to the designer ($M_1 \rightarrow UI_2$).

Designers can refer to a large body of literature on how to improve usability in user interfaces, but the information may not directly influence the usability of modified interfaces; the designer may even make mistakes in connecting the model with design information. To summarize, the first shortcoming is that the steps in the redesign do not always guarantee improvements in usability, as appropriate decisions in the redesign depend on the modeler's judgment. The second shortcoming arises in the step U_2 → M2. To fairly judge usability improvements in a new design, the designer must build a new model. If the usability of the new design is not improved, based on the new model, the whole process of redesign iterates; this is the third shortcoming of the traditional model-based approach.

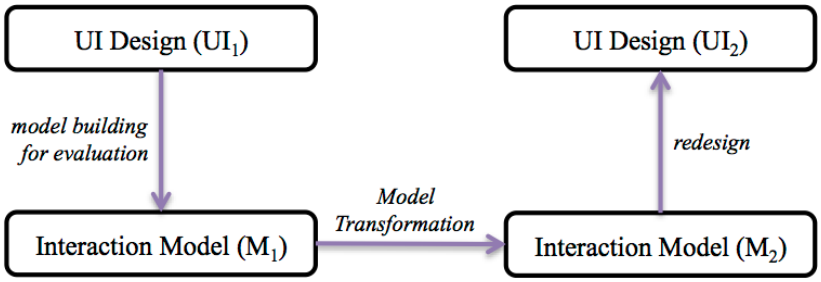


Figure 1.2 Improving design through the model transformation approach

Thus to overcome some of the shortcomings of the traditional model-based approach for usability improvement of a UI, I present a new *model transformation* approach. This approach comprises three steps (Figure 1.2). First, the designer constructs (possibly but not necessarily by hand) a model M_1 to evaluate the usability of a given interface. Second, a set of transformations is automatically applied to M_1 to produce one or more new models (M_2) with improved usability. Third, UI_2 is generated by using design suggestions related to the transformations made in the second step, taking into account the implications of a given model transformation for the design of the interface.

This approach has several potential advantages. Instead of going through repetitive steps for designing, constructing, and evaluating interfaces as in the traditional model based approach, this approach redesigns (transforms) models to give appropriate guidelines to designers, which should lead to improved usability. Because models are more abstract than actual user interfaces, the space of models to consider for improvements is much smaller than the space of interfaces. Further, because model transformations are applied automatically, this approach explores a much larger space of possible improvements than is typically considered by a human designer. This resolves the first shortcoming of the traditional model based approach. Additionally, the transformation from the model of an original interface to another model, then to a new interface, does not require building a new model to validate the usability of new design. Finally, giving appropriate design guidelines that lead to better usability will save designers effort in making their decisions and can potentially reduce the number of redesign iterations.

1.3 Challenges

The goal of this research is to develop a framework to support the model transformation approach for improving the usability of a design during the iterative user interface development process. The first step, constructing an appropriate model from a UI design for usability, is related to model-based evaluation. The third step, recommending or generating a UI design from a model, is related to model-based user interface generation. Past research has shown that both steps are feasible, carried out either by hand or by the system. However, there are no existing approaches for the second step, model transformation. Therefore, this work will focus on what is needed for the model transformation step. With respect to HCI, automating model transformation may reduce designers' cost and effort for UI design. From an AI perspective, developing representations of models for UIs and determining operations for model transformation would make automation possible.

1.4 Contributions

This dissertation makes the following contributions:

- Replication of prior study on model-based usability evaluation: Wilson et al. (Wilson, Mackay, Chi, Bernstein, Russell, & Thimbleby, 2011) emphasize the value of replication in HCI, to ensure that past results generalize, to check assumptions about prior work, and to build new findings on solid ground. The study described in Chapter 3 replicates prior work by John (John, 2011), showing that a modeling tool can be used effectively by novice designers to improve a user interface. My results generalize John's experimental conditions (i.e., the sample of novice designers and the training procedures); the novice designers' work had similar accuracy and consistency. In addition, this result justifies the direction of my dissertation work (Chapter 3).
- A model transformation approach presented and validated through empirical findings: I propose a new approach in the model-based usability evaluation process to help the designers improve usability of UI. I present a framework to support this approach that automatically explores possible alternative designs based on the interaction models specified. The framework also provides recommendations as guidance to designers for usability improvements (Chapter 4). The effectiveness of this approach is validated through empirical findings from two studies: a study that compares the traditional model-based approach and the model transformation approach on a set of design improvement tasks, and an exploratory study on adapting the model transformation approach to the problem of inferring design rationale. From the results of the study, we find evidence that the proposed approach is useful in guiding designers to produce designs with better usability. Also, we see some potential for model transformation approach to be applied to design rationale. (Chapter 6)

- Demonstration of models extending action graphs to represent the model transformation approach: To provide better recommendations to the designer (considering various aspects of the design), I integrate multiple interaction models with action graphs. In theory, the action graph representation is a formalism that can incorporate other interaction models (Thimbleby, 2013). However, a detailed account of how models other than Fitts' Law can be combined with action graphs is missing in the existing literature. I use design information to integrate features of other models into action graphs and show how the usability measures of each model is used for providing the recommendation. (Section 4.1, Section 5.1)
- Implementation of a system reflecting the model transformation approach: Three components are necessary to simulate the proposed model transformation approach: a module to process a *UI representation*, a module to run *model transformations*, and a module to analyze *design rationale*. Each module reflects the conceptual flow of the proposed model approach (Figure 4.1). A module for *UI representation* manages information about the appearance of the design as well as the interaction models of the design. A module for *model transformation* runs a conventional state space search algorithm to explore possible alternative designs represented in successor model states. The state space as well as operator information is collected to analyze design implications in the *design rationale* module. This module generates the recommendation to improve the original design. Implementation of this system also led me to identify important issues for improving the quality of recommendations, such as the need for an automatic layout design algorithm and different levels of abstraction that must be maintained for cost metrics computed by different interaction models (Chapter 5).

1.5 Dissertation Organization

The structure of this dissertation is as follows. In Chapter 2, I present related work on model-based user interface development. In Chapter 3, I present a study that motivates my work. In Chapter 4, the overall process as well as the representation of model transformation approach is described, and in Chapter 5, I present a system designed to execute the process. In Chapter 6, two studies to evaluate my approach are illustrated, with results. Chapter 7 summarizes the contributions and future directions of my work.

2 RELATED WORK

In this chapter, I summarize related work on model-based user interface development. I first go through interaction models for usability evaluation, particularly the models used in my model transformation approach. In addition, an overview of previous model-based user interface development research is described, for relevant ideas about the representation of the model transformation process as well as the content of UI recommendations.

2.1 Model-based User Interface Development

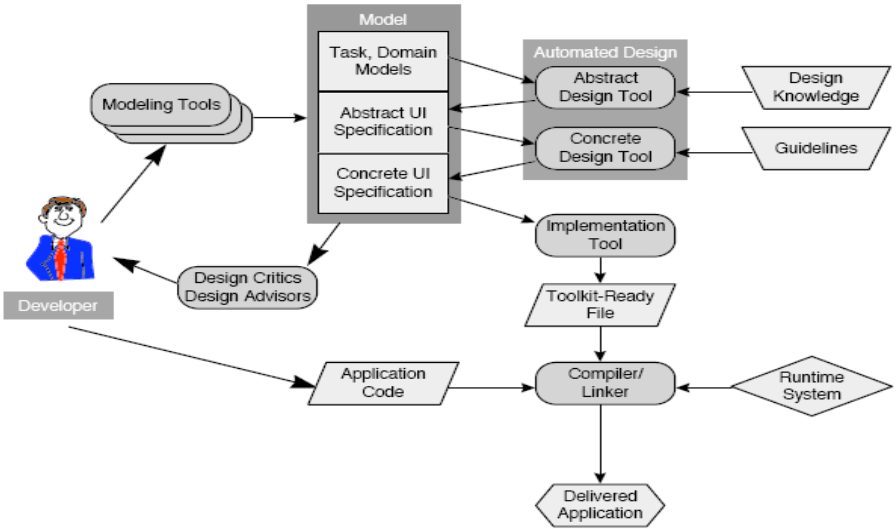


Figure 2.1 Model-based Interface Development Process (Szekely, 1996)

In order to assist in the development of interfaces during the software lifecycle process, a Model-Based Interface Development Environments (MB-IDE) exists. With a model specified for the interface (e.g., data models, task models, or user models), an MB-IDE is used for automatic interface

design generation, expressing design with high-level specifications, help system generation, and generating critics/advisors for a design (Szekely, 1996).

Szekely describes the general architecture for a MB-IDE as shown in (Szekely, 1996). In this architecture, the main components related to my work are the model component and the design critics/design advisors component. The model component has three levels of abstraction: task and domain models, abstract UI specification, and concrete UI specification. A task model specifies the tasks performed using the interface, and a domain model specifies the data and operations supported by the interface. An abstract UI specification represents the structure and content of the interface. It includes abstract interaction objects, information elements, and presentation units. A concrete UI specification specifies the style for rendering in terms of toolkit primitives. The design critics/design advisors component evaluates the interface design. Usually this is done at the level of the concrete UI specification. It suggests refinements based on design knowledge.

2.2 Interaction Design Models

There are a variety of ways to evaluate designs using modeling techniques, including physical models (e.g., Fitts' Law (Fitts, 1954)), performance models (e.g., the Keystroke Level Model or KLM (Card, Moran, & Newell, 1980), (Kieras, 1993), GOMS (Card, Moran, & Newell, 1983), (Gray, John, & Atwood, 1993), (John & Kieras, 1996a), (John & Kieras, 1996b), (Kieras, 1999), ACT-R (Anderson, Bothell, Douglass, Lebiere, & Qin, 2004), (ACT-R Research Group, 2002)), and descriptive models (e.g., state machine models (Dix, Finlay, Abowd, & Beale, 2003)). Each model represents different aspects of UI design, system status, user actions, and performance. In this section, I summarize the interaction design models chosen to integrate into the model transformation system.

2.2.1 Action Graphs

The Action Graph model, due to Thimbleby (Thimbleby, 2013), is a generalized finite state machine formalism. An action graph is a graph in which each vertex represents a state of the system plus the action that leads to that state; edges are transitions between those vertices. It is defined by analyzing all possible state changes based on user actions throughout the design. In other words, the action graph model is a conventional device state representation derived from a system specification with path information encoded in individual vertices and edges. Considering a conventional transition system as G, a vertex $< v, m >$ in action graph G' is interpreted as "vertex v in G is reached by user action m." Thimbleby also describes action graphs as a formalism that could include other user performance models. Unless another model is incorporated to represent user performance, an action graph considers the lower bound estimate of task time as 0.2 seconds times the lower bound count of the action (e.g., button press) to reach the goal.

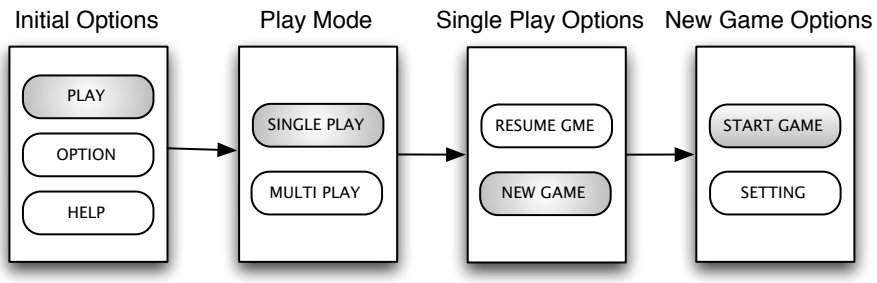


Figure 2.2 Example UI design sequence

As an example, Figure 2.2 shows a sequence of mobile UI screenshots visited in starting a new game, a simplified wire-frame design of the Scrabble[1] interface for the iPhone. Gray rounded rectangles represent buttons touched by a user for this task. Assuming the state is represented as a screen with the user action that caused the transition, the action graph can be represented as follows ('*' is used to represent the action that produced the initial screen, and the name of the state for playing the game is a '*new game')*.

< Initial Options, * > → <Play Mode, Play>
<Play Mode, Play> → <Single Play Options, Single Play>
<Single Play Options, Single Play> → <New Game Options, New Game>
<New Game Options, New Game> → <*new game*, Start Game>

The first line of this model represents the first screen transition in playing Scrabble (Figure 2.2). The <*Initial Options,* *> state represents the initial screen. *Initial Options* is the name of the screen and '*' is the name of the action that corresponds to the transition to the *Initial Options* screen. <*Play Mode, Play*> represents the state of the second screen in Figure 2.2. *Play Mode* is the name of that screen and *Play* is the name of the action (tapping the *Play* button on the *Initial Options* Screen) that leads to the *Play Mode* screen. In sum, the first transition represents the transition caused by tapping the *Play* button on the *Initial Options* Screen to move to the *Play Mode* screen. Each line of this example model represents each transition (shown as arrows between the screens), while the last line represents a transition to the screen for playing the actual game (which is not included in the screen shot sequence).

[1] Scrabble is a mobile game application developed and distributed by Electronic Arts Inc

2.2.2 Keystroke Level Models

The Keystroke-level Model (KLM) (Card, Moran, & Newell, 1980), (Kieras, 1993) decomposes a task into a sequence of operators and predicts user performance. This model considers only primitive operators at the level of pressing buttons and moving the mouse cursor, plus an operator representing mental preparation. These operators are shown in Table 2.1.

Table 2.1 KLM operators and standard execution times (Kieras, 1993)

Operator	*Description*	*Times (secs)*
K	Keystroke. Pressing a key or button on the keyboard	0.12 - 1.2; use 0.28
P	Point with mouse to a target on the display	1.1
B	Press or release mouse button	0.1
H	Home hadns to key board or other device	0.4
D	Drawing n straight-line segments having a total length of l cm	0.9 x n + 16 x l
M	Mentally preparing for executing physical actions	0.6 - 1.35; use 1.2
R(t)	Response of t sec by the system	Must be determined

In order to evaluate usability using the KLM, representative task scenarios for an interface must be chosen. Keystroke-level actions needed for a task are specified, based on the interface design, and the physical operators corresponding to each keystroke-level action are determined (e.g., pressing a mouse button after pressing a keyboard key with the right hand implies a movement of the hand from the keyboard to the mouse). After this step, the physical operators corresponding to the action sequence for performing the given task are listed. Operators representing the user waiting for the system to respond are added, and mental operators (M) are inserted based on heuristics.

User execution time is calculated by summing the standard execution time of the operators. If we assume T_i denotes the product of n_i and t_i, where i is type of operator, n_i is number of operators i, and t_i is standard execution time of i, then the total execution time can be calculated as:

$$T_{execute} = T_K + T_P + T_B + T_H + T_D + T_M + T_{R(t)}$$

Figure 2.3 shows a KLM model shown for the task of finding the file icon to delete and dragging it to the trash can, as given by Kieras (Kieras, 1993).

Operator sequence
1. initiate the deletion **M**
2. find the file icon **M**
3. point to file icon **P**
4. press and hold mouse button **B**
5. drag file icon to trash can icon **P**
6. release mouse button **B**
7. point to original window **P**

Total time = 3**P** + 2**B** + 2**M** = 5.9 sec

Figure 2.3 KLM for 'Find the file icon to be deleted and drag it to the trash can' task (Kieras, 1993)

The KLM was originally proposed for the desktop computer environment. However, one of the benefit of using the KLM is that the operators as well as the time durations can be adapted based on the interaction style specific to the devices. For instance, Holleis et al. (Holleis, Otto, Hussmann, & Schmidt, 2007) present a KLM for the mobile phone that defines and adapts operators for user interaction in the mobile phone UI (i.e., Mobile KLM). The Mobile KLM predicts interaction time on mobile devices. Whereas the original KLM defined six operators and time values for interaction on desktop computers in order to predict the execution time, Mobile KLM basically extends the original KLM to the characteristics of interaction on mobile devices. The Mobile KLM added some new

operators and re-examined the original operators to determine whether they are applicable as is, derivable to new values, or not applicable. The Mobile KLM is used to determine the efficiency of a task in a given interface design.

The Macro Attention Shift (S_{Macro}) operator is used to model the time needed for shifting focus between an object on the screen and in the real world. The Micro Attention Shift (S_{Micro}) operator is used to model the time for looking at the different regions on the device (e.g., keypad, hotkey, screen). The Distraction (X) operator is for times when a user is distracted by some external event; the Action ($A(t)$) operator is for complex actions; the Gesture (G) operator is for the system to recognize gestures; the Finger Movement (F) operator is for the action of moving a finger; and the Initial Act (I) operator is for preparing the mobile device before actually using it. The conventional KLM operators, Keystroke (K), Pointing (P), and Homing (H), are also part of the Mobile KLM. The Mental Act (M) operator and Response time ($R(t)$) operator are used unchanged from original KLM; the Drawing ($D(nD,ID)$) operator is eliminated as not being applicable. Execution time is defined below, where $op=\{A, F, G, H, I, K, M, P, R, S_{Micro}, S_{Macro}\}$.

$$T_{execute} = \sum_{op \in OP} (n_{op} + d_{op} \cdot X_{slight} + D_{op} \cdot X_{strong}) \cdot op$$

Mobile phone technology has evolved since the Mobile KLM was originally developed. Nowadays, mobile phones with touch screens enable users to perform new actions. Therefore some operators of the Mobile KLM may need to be re-examined considering the features of current devices. For instance, the Drawing operator ($D(n_D,I_D)$) is considered not applicable, but it is available on mobile phones with touch screens in a variant form for dragging and dropping. However, re-adapting and validating the duration of each Mobile KLM operator is not in the scope of this dissertation. In this dissertation, I assume the operators and time duration for Mobile KLM in their current form.

2.2.3 Fitts' Law

Fitts' Law is one of the most commonly used physical interaction design models; physical models do not consider purely cognitive activity. Fitts' Law analyzes the time for tapping a target based on the size of the target and the distance to it. This model is defined under the assumption that information capacity and the human motor system are related (Fitts, 1954).

Fitts' Law (Fitts, 1954) is defined by three equations. These equations define the index of difficulty (*ID*), movement time (*MT*), and index of performance (*IP*), also called throughput (*TP*). Together these equations are commonly used as the measure of the performance of an input device.

ID evaluates the difficulty of a target selection task in bits, based on the width of the target and its distance from a known starting point. The formula is as follows, where *A* is the distance to the target and *W* is the width of the target.

$$ID = log_2(\frac{A}{W} + 1)$$

MT evaluates the time to complete a task based on *ID* and two empirically derived coefficients based on experimental conditions. The equation is as follows. The constants *a* (in units of time) and *b* (time units/bit) are coefficients that represent the efficiency of the device.

$$MT = a + b \times ID = a + b \times log_2(\frac{A}{W} + 1)$$

IP (or *TP*) represents human performance capacity and is defined as the quotient of *ID* and *MT*.

$$IP = \frac{ID}{MT}$$

The design implication of Fitts' Law is that it can be used as a strategy for improving usability of a user interface (Heim, 2007). A common summary is that 'Large targets and small distances between targets are better for usability'. This is because such constraints on targets will reduce the index of difficulty.

2.2.4 Parhi et al.'s Model

Parhi et al. (Parhi, Karlson, & Bederson, 2006) present models relating target size to performance for one-handed thumb use on touchscreen devices. For their models, they conducted two experiments. One was to determine appropriate target sizes for the task of pointing at a single target and the other was to determine appropriate target sizes for the task of sequential pointing to multiple targets.

By analyzing the accuracy of a touch based on the target size from their experiment, the following models were presented.

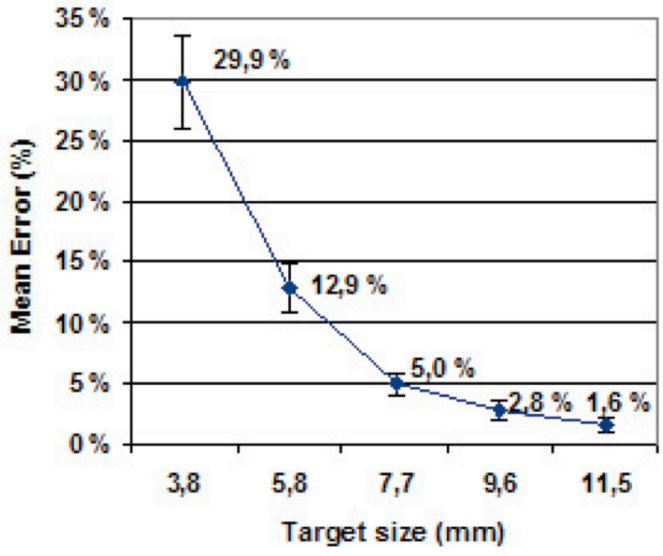


Figure 2.4 Mean percentage of erroneous trials to single target for each target size

The single-target task represents typical interactions with radio buttons, activation buttons, or checkboxes. Figure 2.4 shows the result of the study measuring error rate versus different target sizes (Parhi, Karlson, & Bederson, 2006). From this study, targets smaller than or equal to 7.7 mm resulted in significantly more errors than targets larger than or equal to 9.6 mm. There was no significant difference in error rates for the targets larger than 9.6 mm.

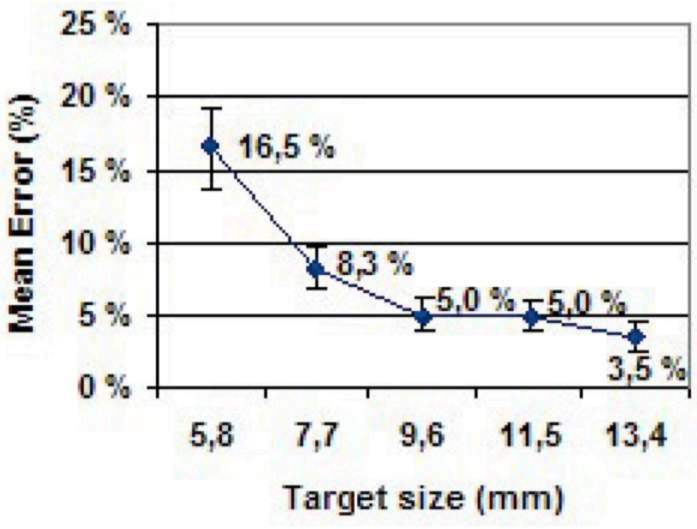


Figure 2.5 Mean percentage of erroneous trials to multiple targets for each target size

The multiple-target task represents those similar to numeric or data entry, such as pressing a series of numbers on a keypad. Figure 2.5 shows the result of the study of measuring the error rate of this task on multiple targets versus different target sizes. The targets with smaller sizes had a larger error rate, while there was no significant difference in error rates between targets with size larger than or equal to 9.6 mm.

2.3 Model-based Usability Evaluation Tools

Sears (Sears, 1995) proposed AIDE, a metric-based interface design and evaluation tool that extends the idea of Layout Appropriateness (Sears, 1993). In his earlier work, Sears presented a metric to measure how far a proposed layout of widgets is from an optimal layout. This metric is based on frequency of use and the distance involved in performing the task. The optimal layout is constructed by considering the interface as a collection of unit-sized objects placed on a rectangular grid and then finding an optimal solution by searching through possible layouts. AIDE defines the metric presented in (Sears A. , 1993) as efficiency. In addition, it defines metrics for alignment, balance, and other constraints. AIDE evaluates these measurements for a given UI interactively. Byrne et al. (Byrne, 1994) presented a UI in a user interface development tool, called USAGE. This tool incorporates a formal interface analysis technique (GOMS) in a model-based interface design tool (UIDE). A translator was built to translate the action sequence files of UIDE into a NGOMSL (Natural GOMS Language) (Kieras, 1988) model, so that USAGE provides execution time estimates based on the GOMS model of the generated UI. A further application of USAGE is in supporting a designer to run alternative UIs to quickly compare interfaces.

2.4 Automatic Interface Generation

2.4.1 Model-based Interface Generation

Puerta et al. (Puerta A. R., 1994) present a model-based UI development tool called Mecano. This tool uses a data model and a high-level dialog specification as a model, and generates a UI using design templates and layout rules. Szekely et al. (Szekely, Luo, & Neches, 1993) present HUMANOID, which takes a model as a declarative description of the interface presentation and behavior. It creates a UI based on templates and the result is shown at runtime to assist the designer.

2.4.2 Interface Generation with Adaptation

SUPPLE (Gajos, 2004) is a system that generates an interface meeting the device's constraints and minimizing user's effort on using the interface. This system considers the interface generation problem as an optimization problem. Gajos formally defined device characteristics by a functional interface specification and device capabilities, then modeled users' usage patterns from traces of performance. When rendering an interface, a branch-and-bound constrained search algorithm is applied to find an optimal solution. A different interface design is rendered adapting the characteristics of the device as well as the usage pattern.

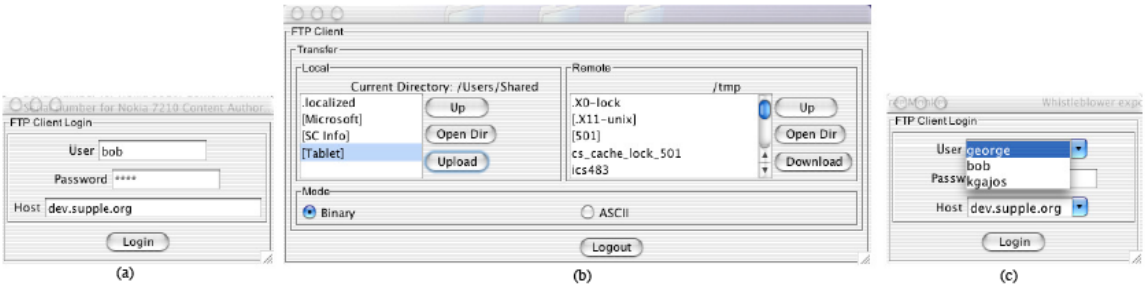


Figure 2.6 Interfaces for the FTP client application (Gajos, 2004)

Figure 2.6 shows a set of interfaces for the FTP application. Screenshots (a) and (b) are the initial designs with the following usage pattern: (1) if the information entered in (a) is correct and the Login button is clicked, the window changes to (b); (2) if the Logout button in (b) is clicked, the window changes to (a). With this information, a new design (c) is created using SUPPLE.

2.5 Tools for Interface Guidelines

Just as there is a vast number of guidelines in the form of recommendations, platform-specific style guides, and experience-based usability heuristics, there is a great deal of research on tools for working with guidelines (Grammenos, 2000). Vanderdonckt's work on a system to manage those guidelines, called SIERRAS, is one such tool (Vanderdonckt, 1995). However, this system focuses on managing guidelines rather than on application to real interface designs. Sherlock (Grammenos, 2000) is a system that provides an integrated environment for using guidelines. It automatically does a usability inspection of a preliminary design, based on given rule-based guidelines. The role of my system is similar to Sherlock in the aspect of analyzing a user interface to improve its usability. However, it is different in that Sherlock is using UI guidelines whereas my system is using interaction models for analyzing the interface and giving guidance on how to improve its usability.

2.6 Discussion

Work on model-based user interface development continues today; however, there are fewer connections between automatic interface design generation (i.e., automatically generating user interfaces, tools for interface generation) and approaches that incorporate design critics/advisors (i.e., interaction design models, model-based usability evaluation tools). Work on interaction models and model-based usability evaluation tools focuses on models to better represent user performance. Work on automatic interface design generation focuses on models to better represent interface designs and to better mange interface guidelines. The model transformation approach connects the research in these two areas. In my work, interface usability is analyzed using interaction design models or using model-based usability evaluation tools. I associate results from the models with relevant design information. Through this relationship, my approach provides recommendations or guidelines to

improve the usability of an interface. This can be further used to automatically generate user interfaces.

3 MODELING TECHNIQUES FOR UI RECOMMENDATIONS[2]

In this chapter, I present two studies on the use of a modeling tool for making design recommendations. The results of this study motivate the use of model transformation approach in the UI development process.

3.1 Background

Extensive research in the field of interaction models for usability evaluation has been done, producing metrics for re-design decisions or techniques to compare usability between different designs. However, it is rare to find work dealing with the issue of bridging model information to UI recommendations.

In 2011, John (John , 2011) reported a study of 100 novice modelers who used CogTool[3] to produce recommendations for improving two Web sites. Modelers showed significant consistency in their results, and the majority of their recommendations were well supported with quantitative justifications based on CogTool models. The focus of this paper was showing that CogTool can be easily learned and applied by those new to modeling and relatively new to interface design, whereas modeling tools are typically viewed as helpful mainly to expert modelers in understanding user performance. In addition to this result, this work was closely related to my view on the role of modeling tool: the scope of the study was analyzing the use of the tool for producing UI

[2] This chapter is an edited version of a paper published in the proceedings of the ACM Conference on Human Factors in Computing Systems, or CHI (Hong & St. Amant, 2014).
[3] CogTool (John, Prevas, Salvucci, & Koedinger, 2004) (John & Salvucci, 2005), is a cognitive modeling tool for HCI; it predicts human performance for storyboarded user interfaces. In the past few years it has been applied in real-world software development projects to good effect (Bellamy, John, & Kogan, 2011).

recommendations, which normally requires the user to understand the model and to link that knowledge to both the design and the task.

3.2 Experiment

As the ultimate goal of this dissertation work is to provide a system that helps designers by providing usability evaluation measures as well as UI recommendations, John's study is good motivation. We conducted two follow-on studies. The first was a simple replication of John's study. The purpose of the study was to determine whether John's findings would generalize beyond the sample of novice modelers at CMU, taught to use CogTool by one of its developers. We investigated whether model information (i.e., CogTool results) help in producing design recommendations, using a different population of novice designers/modelers. CogTool shares the goals of other task modeling techniques for HCI (e.g., the Keystroke Level Model (Card, Moran, & Newell, 1980)), but we are aware of no empirical comparison between CogTool and these techniques for making design recommendations previous to my own work.

The second study was a follow-on in which modelers, without knowledge of CogTool but with the same exposure as in Study 1 to the KLM and other techniques, recommended improvements to the same user interface. The purpose of the second study was to test whether different modeling techniques impact performance in producing design recommendations.

3.2.1 Participants

Undergraduate computer science students in an HCI course participated in the studies. The course is a conventional survey of topics in HCI, using *The Resonant Interface* (Heim, 2007) as the textbook. At the time of each study, modeling topics in class had covered Fitts' Law, KLM, GOMS, and modeling

for HCI in general (three 1.25-hour lectures in all). Students submitted models as part of a homework assignment, worth 5% of their course grade.

3.2.2 Study 1: Recommendations with CogTool

Participants were given an explicit task to perform on a Web site, shown in Figure 3.1. We chose a simple task comparable to John's for our findings. Figure 3.2 shows the design sequence mapping to the task instructions in Figure 3.1, with the area where the task was performed marked in orange rounded rectangles.

Participants were also given instructions for building a model for the task and making three recommendations for improving the interface. Additional instructions directed participants to carry out the task by hand first, to be as specific as possible in their recommendations, and to remember that (a) the goal was to model expert behavior and (b) not all potential changes, such as reducing visual clutter or changing the color scheme, would plausibly improve expert behavior, and such changes were not amenable to analysis using the techniques they were to rely on.

1. *Visit the Web page http://www.amazon.com. (If you are automatically logged into your Amazon account, log out, then reload the page.)*
2. *Click in the Search text box.*
3. *Type "kinect" (without quotation marks) in the Search text box, and press return.*
4. *Click on the top link that is shown in the search results; for me this is "Kinect Sensor with Kinect Adventures and Gunstringer Token Code."*
5. *Click on the Add to Cart button on the right.*

Figure 3.1 Task instructions given to students

Figure 3.2 Screen shot sequence mapping to task instructions

Determine three potential improvements to the usability of the Web site, for this task, based on the information in the visualization. For each of your improvements,

1. *Give a short phrase that identifies the improvement.*
2. *Optional: If the phrase doesn't entirely specify what the change to the Web site should be, explain it in more detail.*
3. *Give a screenshot of the visualization, annotated to show how the improvement would change performance (e.g., a circle around some pattern).*
4. *In words, summarize the pattern you've identified in the visualization.*
5. *Give the time savings, in seconds and milliseconds, that your improvement would produce.*

Figure 3.3 Modeling instructions given to students

Figure 3.3 is the set of modeling instructions for Study 1: participants were to build their models in CogTool and support their recommendations with annotations of a CogTool timeline visualization. Study 1 replicates John's as follows: the participants were new to modeling and CogTool, they were given a step-by-step description of a task, and they completed their modeling and recommendation activities within one week. The instructions were equivalent to John's, though more explicit in describing the required structure for submissions. There were differences. The participants were all computer science majors rather than HCI majors with a mix of backgrounds, and they were asked to analyze a different Web site from the ones used by John. The introduction to CogTool, including a demonstration, took an additional half hour of class time. The introduction was given by someone not on the CogTool development team, and unlike John's study no hands-on working session followed during which questions could be answered. (The instructor answered a few procedural questions asked by the participants via email.) Otherwise Study 1 was as close to John's as practicable.

3.2.3 Study 2: Recommendations with Other Modeling Techniques

Study 2 was conducted as one semester later in the same course. The conditions in Study 2 were the same as in Study 1, except that the modeling task was given to students *before* the introduction of CogTool. (They used CogTool later in the course for their project work.) The modeling instructions for Study 2 gave more generic guidelines, without mention of CogTool: *Base your structured description on the material in Chapter 7 and our discussion in class: the Model Human Processor, with its visual, motor, and cognitive capabilities; the Keystroke Level Model; various GOMS techniques*. Modelers worked without specialized tools, mostly relying on Fitts' Law and KLM. Aside from the use of CogTool, Study 1 and Study 2 are effectively identical, with respect to the task and the background of the participants.

3.3 Results

3.3.1 Study 1: Recommendations with CogTool

In Study 1, 53 modelers submitted recommendations. Two submissions were produced without using CogTool and are not included in our analysis. Of the remaining 51, two modelers provided only two recommendations rather than three, giving 151 recommendations in all. Figure 3.4 shows CogTool model submitted by one of the student. The right side of the model presents the steps of the task, and left presents the portion of the design for the selected step of the task. Predicted task execution times will appear when the Compute button is clicked. Figure 3.5 shows the CogTool timeline visualization of the model marked with the portion of the model that supports the UI recommendation the student has made.

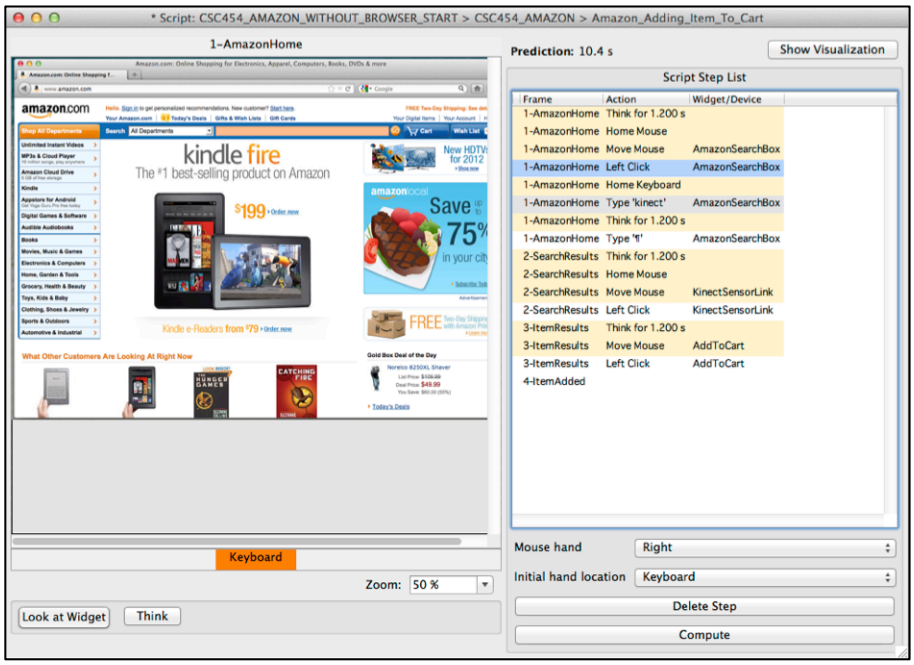


Figure 3.4 CogTool model for Amazon task

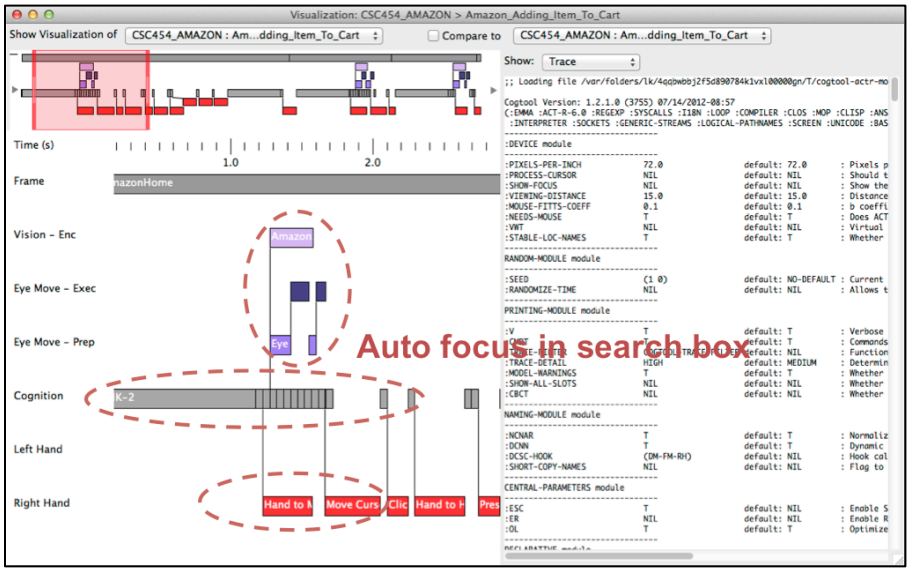


Figure 3.5 CogTool timeline visualization with annotation to support recommendations

To analyze results, two evaluators independently created categorizations over the aggregate set of recommendations.[4] The evaluators then worked together to agree on a specific category for each recommendation. As shown in Table 3.1, the most common recommendations were to place an additional "Add to cart" button beside each product and to make targets for mouse clicks larger or to place them closer together.

Following John, the recommendations were judged with respect to being "correct" (whether they would reduce the predicted duration of the task) and "well supported" (by reference to a visualization produced by CogTool). The evaluators judged 68.2% of the recommendations correct and well

[4] Our emphasis is not on the categorization of novice design recommendations, but we note the challenge of consistent categorization of usability improvements based on textual descriptions and CogTool visualizations, working without an *a priori* set. The two evaluators (with both research and practical experience in HCI and usability) differed in the number and assignment of categories they identified, due to ambiguities or combinations of categories in the students' descriptions. There was initial agreement on 74% of the recommendations, with Kappa = 0.693.

supported (c+ws), below John's 75.1%. An additional 19.9% were correct but "vague or inarticulately argued," in most cases because the explanation did not refer to the model (cf. John's 16.6%). The remainder were incorrect. In total, 88.1% of the recommendations were judged to be correct, if not necessarily well supported (cf. 91.7%).

Table 3.1 Study 1: Correct and well-supported recommendations

Recommendation	*# Participants (%)*	*Median*	*IQR*
"Add to cart" button for each product	25 (49 %)	1.93	0.22
Closer/larger targets	22 (43 %)	0.18	0.35
Incremental search, per-keystroke updates	17 (33 %)	1.67	0.72
Autofocus in search box	15 (29 %)	2.54	1.25
"I feel lucky" button	4 (8 %)	1.89	1
Keyboard navigation versus mouse clicks	4 (8 %)	0.57	3.4
Search autocomplete	4 (8 %)	0.7	0.55

Table 3.1 shows all of the correct and well-supported recommendations given by participants; no others were identified. All but one modeler (98%; cf. John's 100%) gave at least one correct recommendation. 42 of the 51 gave at least one correct and well-supported recommendation (82%; cf. John's 96%—possibly due to the abbreviated instructions for CogTool in Study 1). Fourteen participants gave one or more incorrect recommendations (27%; cf. John's 21%). These represent poorer performance than John's numbers, but most measures are surprisingly close.

Table 3.1 also shows the median duration and interquartile range of the time savings in seconds calculated by the participants, per category of recommendation. The total time for the task is more than ten seconds, and most of the recommendations would each save half a second or more. We find it notable that participants can justify recommendations, while they lacked some information to make contextually-appropriate recommendations, for improving a professionally designed interface in wide use.

Variability in estimates of time savings (in the IQR column) is inevitable, due to the coarseness of categorization. For example, changing the size or location of a target icon can reduce duration, but how much depends on the new size or location; *Search autocomplete* and *Incremental search* can reduce typing time, but the reduction depends on how many keystrokes are saved. Some but not all participants modified their models for these two design changes; others estimated the savings by subtracting some number of keystrokes. The latter approach appeared to be common, but it will not always produce accurate results. *Incremental search* savings are inflated because most participants did not include the required shift of visual attention to a dynamically changing set of products. Even for more straightforward recommendations, such as *Autofocus in search box*, participants differed in whether the *Think* operator should be included in the savings. As long as the recommendation was correct and the annotation included the relevant cognitive/visual/motor components of the task, it was considered well supported.

One category of interest not included in Table 3.1 is *Remove extraneous content*. Many participants observed that removing clutter could improve performance by allowing for target elements to be closer together (leading to a *Closer/larger targets* categorization), but other participants recommended reducing clutter based on incorrect arguments (e.g., that experts are distracted by a cluttered visual display, or that thinking time is spent processing screen information not included in a model). These were considered incorrect.

In general, incorrect recommendations were of two types: within the scope of the task but not likely to improve an expert user's performance (e.g., removing ads or changing button colors—8% of all recommendations); or out of scope, without regard for expertise (e.g., providing more results to scroll through—4% of all recommendations).

We evaluated the consistency of the recommendations with the Any-Two Agreement (A2A) statistic (Hertzum & Jacobsen, 2001). A2A is the number of common problems found by two participants, divided by the total number of problems found by those two participants, averaged over all participants. John's A2A value of 34% puts her study above 9 of the 12 studies in Hertzum and Jacobsen's survey; our A2A value of 30% is above 8. Study 1 adds evidence to the view that CogTool may reduce the evaluator effect by providing a model-based grounding for recommendations (John , 2011).

Two weeks after the completion of the modeling exercise a brief, informal survey was given. 47 students submitted free-form text answers to questions about their use of CogTool. (The survey was anonymous to allow the question, *Did you follow the instructions?*; all respondents answered yes.) The most informative answers were to the question, *Describe in a few sentences how CogTool fit into the process of your making recommendations for design improvements.* John has demonstrated that novice modelers can produce useful results with CogTool; our survey provides initial clues about how they integrate modeling into their analysis, as well as how CogTool improves the process.

Our informal analysis of the responses showed three categories of interest. The most common answer, from 26 participants (51%), described using CogTool to identify appropriate areas for design changes. "I would use CogTool to see which areas of an application take the longest for a user to move through." "I looked at the model that CogTool presented and looked for areas where there were actions canceling each other out, such as moving a hand to the mouse and then back… and tried to understand how to eliminate some of those steps." Most then described using CogTool to justify their recommendations in quantitative terms. Nine participants (18%) also mentioned making comparisons between models, to evaluate the differences between interface designs. "I then looked at side-by-side comparisons… to see where differences existed." Thirteen modelers (25%) only mentioned using

CogTool to validate design decisions they had thought of on their own: “It helped give me proof that my suggestions would actually help by providing estimated times.”

3.3.2 Study 2: Comparative Analysis

For Study 2, in which participants did not use CogTool, 50 participants submitted 150 recommendations in total. Figure 3.6 shows a Keystroke Level Model built by one student. We have marked the portion of the model the student pointed out to support the recommendation. This student said that suggested recommendations will remove the marked operator sequences, so as to improve total execution time of the task.

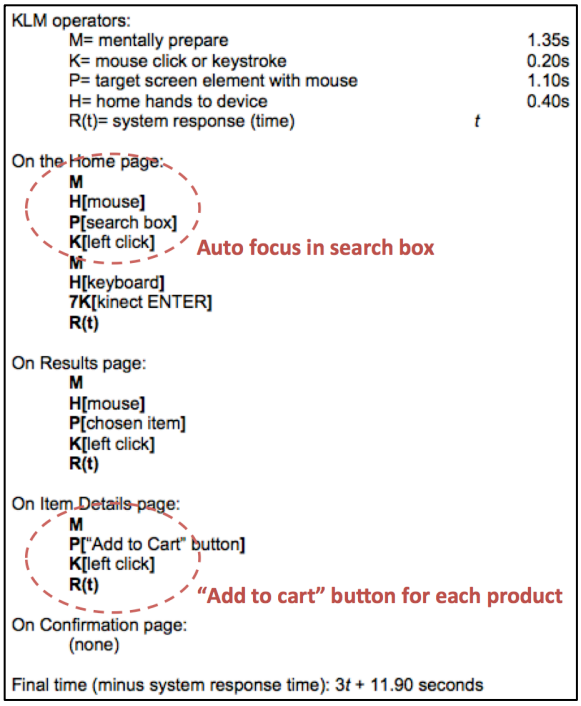


Figure 3.6 KLM with mark-up support recommendations

The same evaluators followed the same procedure as in Study 1 to categorize recommendations, except using references to the modeling techniques above rather than to CogTool in their evaluation.

Table 3.2 Study2: Correct and well-supported recommendations

Recommendation	*# Participants (%)*	*Median*	*IQR*
"Add to cart" button for each product	18 (36 %)	2.33	1.3
Closer/larger targets	8 (16 %)	0.7	0.86
Incremental search, per-keystroke updates	10 (20 %)	0.75	1.83
Autofocus in search box	20 (40 %)	1.64	1.23
"I feel lucky" button	5 (8 %)	2.01	2.38
Keyboard navigation versus mouse clicks	3 (6 %)	0.93	3.57
Search autocomplete	2 (4 %)	0.26	0.28

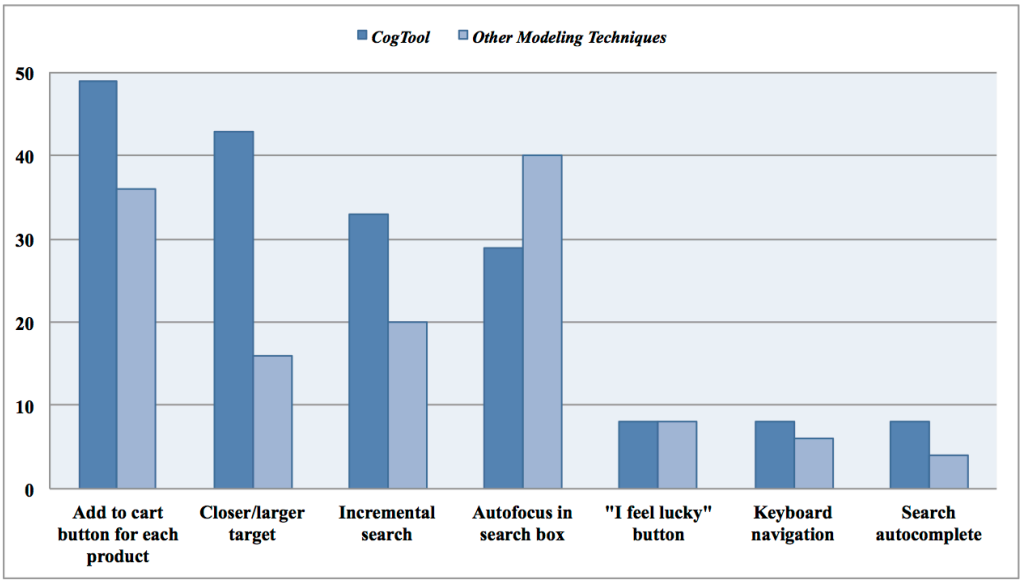


Figure 3.7 Percent of participants per recommendation category

Table 3.2 shows that the correct and well-supported recommendations in Study 2 and Figure 3.7 shows the numerical distribution of categories and estimates of time savings comparable to those of Study 1. But the totals were considerably lower. In Study 1, 88% of recommendations were correct, versus 63% in Study 2; for correct and well-supported recommendations the values were 68% and 43%.

Table 3.3 Performance per participant as tail distribution function: % of participants

Performance	*Study 1*		*Study 2*	
	correct (%)	*correct & well-supported (%)*	*correct (%)*	*correct & well-supported (%)*
# of recommendations = 3	69%	49%	40%	28%
# of recommendations ≥ 2	94%	73%	62%	48%
# of recommendations ≥ 1	98%	82%	88%	54%

Table 3.3 shows the tail distributions for the count of correct/well-supported recommendations, aggregated over modelers; that is, 35/51 or 69% of modelers in Study 1 gave three correct recommendations, and so forth. A two-sample Kolmogorov-Smirnov exact test shows a significant difference between Study 1 and Study 2: $D = 0.321$, $p < 0.01$. (Treating the counts as continuous data, we can ask about the difference between the mean number of correct recommendations per modeler, 2.6 in Study 1 versus 1.9 in Study 2: $t(99) = 3.97$, $p < 0.01$.) Analysis of correct and well-supported recommendations shows the same result. In other words, performance in these studies is not determined purely by the task and the background knowledge of modelers; use of CogTool is a significant factor. In general, this is evidence that using different modeling techniques may influence performance.

A2A, a measurement about the extent on evaluators agreement for the problems of the system (Hertzum & Jacobsen, 2001), in Study 2 is 20%. The drop from 30% in Study 1 is not unexpected, given the greater variation in modeling procedures. No new correct and well-supported recommendations were identified, but there was a wider range of incorrect recommendations. These fell into the same general classes as in Study 1: changes affecting performance outside the scope of the task (e.g., multiple item selection or filtering capabilities—11% of all recommendations), and not focusing on expert users as specified in the instructions (25% of all recommendations). The latter is not surprising. In an HCI survey course, the clearest illustrations of usability issues are often based on the challenges that novice users face in complex interfaces; CogTool currently targets expert users. The former category of incorrect recommendations is more surprising, though not in retrospect. CogTool also appears to help novice modelers remain aware of the boundaries of a specific task when attempting to improve user performance by changing the UI design.

3.4 Discussion

Although these two studies were designed with a focus on whether novice modelers would be able to make and support their design recommendation using CogTool, and whether CogTool was a decisive factor, the results also motivate and provide direction for my model transformation approach of using interaction models for producing better designs.

Study 1, conducted with students (i.e., novice modelers) at a different university, gave an interface design with task to it, build a model using CogTool, make design recommendations, and support recommendations using CogTool. The result shows that students can produce and justify design recommendations for improving a specific user interface with similar accuracy and consistency as in John's study. This supports the possibility of using CogTool results to identify UI problems and make

design recommendations, not limited to just comparing execution time predictions for alternative designs. This adds to my motivation for providing a framework that automatically identifies UI problems as well as produces design recommendation, through the model transformation approach.

Study 2 evaluated performance using different modeling techniques. Relatively large differences in performance were observed, compared with Study 1. This indicates that CogTool influenced users in their ability to detect UI problems and produce good re-design recommendations to resolve the problems. Also, in general, it indicates that performance can be different based on which modeling technique they use. In practice, each modeling technique focuses on quantifying a different aspect of user behavior or the design. Design principles or design guidelines can be of more or less use based on the design or the domain of the application. This led me to incorporate multiple modeling techniques in my approach.

Existing research provides limited information about how modeling can best be integrated into interface design tasks (e.g., (Bellamy, John, & Kogan, 2011)), but the strategies used in Study 1 are among those typically recommended by experienced HCI modelers. Moreover, recognized by one of the students answer from the survey at the end of Study 1, "I would never have thought about these areas taking up users' time, so CogTool did help…" our approach seemed to be in the right direction.

3.5 Limitations

Although the studies described in this chapter provide motivation for the model transformation approach, the study design has limitations. Study 1 was designed to replicate John's study. We were able to show similar accuracy and consistency in modeler performance. However, the interface and the task were not the same as in John's study (also, John used two interfaces and our study used one interface). Our conclusion was that John's results generalized beyond her sample of modelers, but a

description of the differences between the interfaces as well as the task would have added more evidence for the success of the replication. In the same vein, more detailed information about the materials given to modelers before the study as well as the lectures related to CogTool should be considered as well. Also, in order to convincingly argue that the CogTool can be used for identifying usability problems and supporting related recommendation by novice modelers in general, more studies using other interfaces, including more complex interfaces with more complex tasks, are necessary. Such work is already underway (Bellamy, John, & Kogan, 2011). Study 2 was conducted a semester later than Study 1. However, there were differences in the modelers' knowledge between these samples, particularly related to knowledge of CogTool. Study 2 was conducted before the introduction of CogTool. Although the tasks of Study 2 did not require knowledge of CogTool, participants may have performed differently if they had been exposed to CogTool, in that the system is a specific, detailed example of general modeling techniques.

4 THE MODEL TRANSFORMATION APPROACH

In this chapter, I investigate how a UI recommendation system supporting model transformation can be designed to overcome the shortcomings of the traditional model-based UI development process.

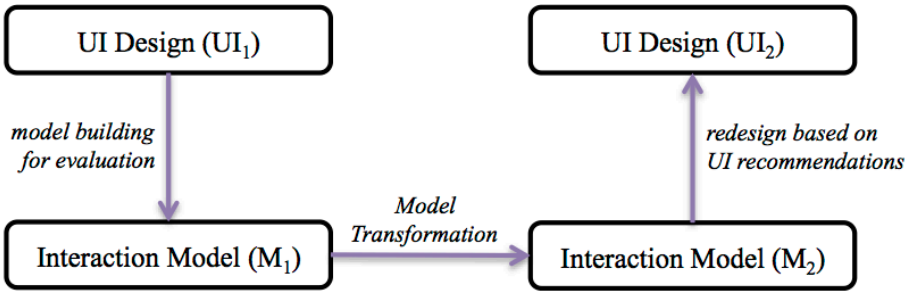


Figure 4.1 Improving design through the model transformation approach

Through the *Model Transformation* approach, by considering a set of transformations applied automatically to the model, the designer can directly see and apply recommendations for redesigning a UI with improved usability. This approach has several features that can potentially improve the UI development process because (1) models are more abstract than an actual user interface, and thus the space of models is much smaller than the space of interface design to consider, and (2) information generated by transforming models will contribute to design recommendations, therefore directly guiding the designer to improve the usability of the original design. In other words, detecting potential usability problems and analyzing design implications for solving usability problems could be automated to a significant extent. Currently this is the responsibility of the designer, in the traditional approach to interface design. Further, this approach reduces the iterative steps for redesign that may be due to incorrect design decisions (i.e., no improvement in usability).

To achieve model transformation approach, I consider the model transformation step as a problem of state space search. Conceptually, a *model state* represents the set of interaction models (which analyze the usability of the design) with a corresponding UI design and user task information. The initial state is a model state of the original design. An *operator* represents a transformation rule that operates on a model state to generate new model state. This operator results in a change to the design as well as the usability. Rather than setting a goal state, I define a goal usability cost, which is a quantitative usability measure produced by the interaction models. Unless there is a specific goal usability cost, my search algorithm will explore all possible successor states based on the operators defined. The purpose of this algorithm is not to identify an optimal new design but to provide plausible guidelines (or recommendations) to improve the usability of the current design. By automatically exploring search space of possible design changes, this algorithm is able to collect usability information of successor model states and able to provide design guidelines (or recommendations) to reach the design for each such state. This will help the designer to change the original design into a new one with better usability.

Once the search algorithm has finished, all *successor model states* represent possible interaction models of possible new designs (M_2 in Figure 4.1). Because each *model state* includes usability measures, and because the goal of this process is to improve the usability of the original design, the designer can make redesign decisions by examining the usability of successor model states. Although operators are defined in model terms, each can be interpreted as an operation on a design component. Therefore, once a successor model is selected, design recommendations (steps of design changes presented at the design component level) to change the original design into the new design can be easily generated by tracing the sequence of operators from initial model state to selected model state.

From the designers' perspective, they will see redesign guidelines that correspond to the usability level they would like to achieve.

4.1 Model States

A model state represents a set of interaction models that evaluate the usability of a UI design. I used action graphs (Thimbleby, 2013) to incorporate other interaction models in the *model state*. Thimbleby describes action graphs as a formalism that could include other user performance models; however, they have some limitations for the purposes of representing model transformation approach.

First, it isn't immediately obvious how to extend an action graph to embed other models. Specifically, when embedding another model in the action graph, representing correlated information between models is not explicit in the formalism. For example, an action graph represents a state of the device, the action for a state transition, and a cost function for the transition. However, the KLM is defined with operators representing physical and mental operators with its sequences. In theory, mapping between the components of two models should be possible, but detailed demonstration was left as future work (Thimbleby, 2013). Second, it isn't explicitly how to represent features related to the design specification. Issues of combining information other than performance metrics with an action graph have not yet been explored in detail.

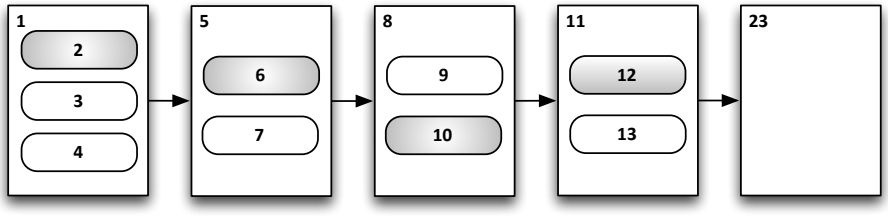


Figure 4.2 Example wireframe design for design components model

Inspired by the fact that all interaction models implicitly or explicitly include information about a given design, I integrated other interaction models into the action graph representation using a *design components model*. A design components model here is defined as information about design components (e.g., buttons, icons, screen, etc.) constituting the design. Figure 4.2 shows a wireframe of an interface (five rectangles represent each screen, rounded rectangles represent buttons, gray rounded rectangles represent buttons actually touched, and arrows represents screen transitions) used for representing design components model for the model state shown in Figure 4.3. When representing interaction models for a model state, I specify a corresponding design component for each model operator or parameter[5].

In the following sections, I list the interaction models I have integrated into model states and describe how each is mapped to design information. More details on the representation will be described in Chapter 5.

Action Graphs

Action graphs are used to measure efficiency. Specifically, an action graphs can be used to analyze lower bounds on task execution time by counting the number of actions in a task sequence. In addition to using action graphs to measure the usability of a design, I used this formalism to incorporate other interaction models.

[5] The initial idea of this representation was presented as a poster at the Graphics Interface conference, GI. (Hong & St.Amant, 2013)

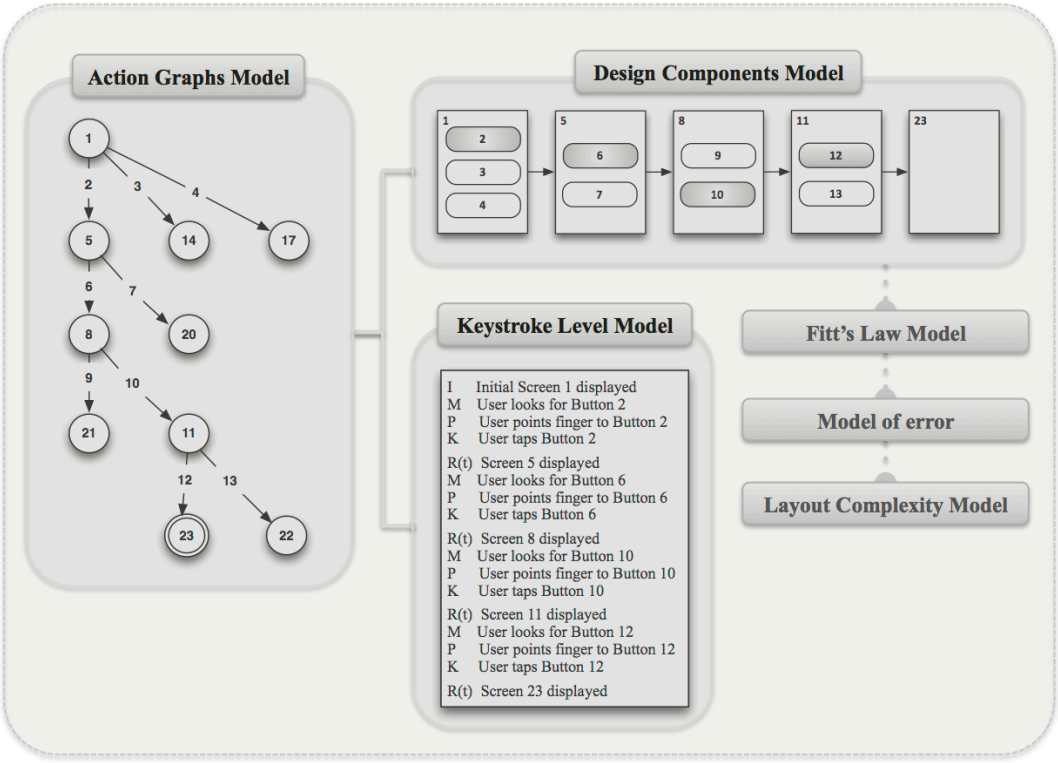


Figure 4.3 A model state composed of interaction models

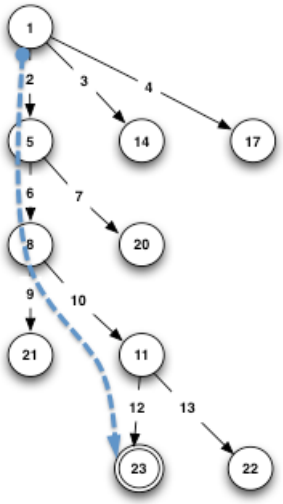


Figure 4.4 Example action graph

To associate action graphs and design components, we assume that the vertex (or state) of a graph representing the state of a device maps to a screen in the design. Also, as the edge (or action) of a graph represents a user action that leads to a state, we consider an edge to be a user action accepted by a design component (e.g., a button). Given my choice of action graphs as a basic formalism, I extend the representation to include design information (i.e., the design components model) as captured by different interaction models. An example of action graph representing part of the design and task of Figure 4.2 is shown in Figure 4.4.

Once an action graphs is defined for the design in a model state, a *goal path* is searched for through the action graph (marked with blue dashed arrow in Figure 4.4). I define a goal path as a shortest path in an action graph from a vertex representing the design of the initial screen (i.e., the path for lower bound action count) to a vertex representing the design of most frequently (popularly) accessed screen (e.g., a screen for playing a new game) by the user.

Keystroke Level Model (KLM)

The KLM is a method for estimating task execution time by analyzing sequence of actions using a specific set of abstract operators (Card, Moran, & Newell, 1980) (Kieras, 1993), as discussed in Chapter 2. For model transformation approach, I redefined two KLM operators: *K* as the screen touch/tap operator and *P* as the finger movement on a screen, with same operator durations.

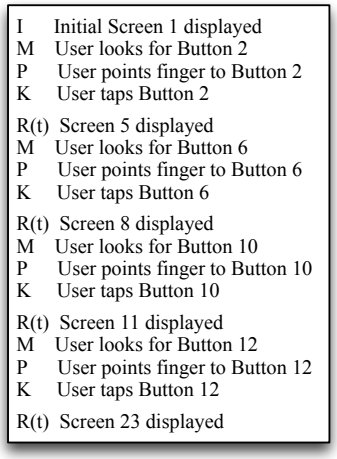


Figure 4.5 Example KLM

To represent a KLM in a model state, I mark the design components corresponding to each KLM operator. For example, if the user is looking for Button 1, I associate Button 1 with a corresponding KLM operator *M (an operator for mental act of routine thinking or perception)*. In the action graph formalism, the KLM operator *R(t)* for system response time (due to a screen transition) is considered a component of an action graph vertex, and operators representing a user action for a screen transition (e.g., *K, P, M*) are mapped to action graph edges.

Larger task execution times (or repetition of some KLM operator sequences) can indicate the desirability of removing a button or a screen to reduce cost.

Fitts' Law

Fitts' Law (Fitts, 1954) is one of the most commonly used physical interaction design models. The Fitts' Law formula for analyzing the movement time to a target is also integrated into the model state representation. Movement time is calculated with the size of the target and the distance to it, using

MacKenzie's SMALLER-OF technique (MacKenzie & Buxton, 1992), where *A* is the distance to the target and *W* is the width of the target:

$$MT = 230 + 166\ log_2\ (\frac{A}{W} + 1)$$

For example, if we calculate movement time of finger movement from the first screen to the second screen shown in Figure 4.2, the movement time based on above equation is 396 ms (assuming the distance is 21 pixels and the width is 21 pixels).

Whereas action graphs, the KLM, and design components are transformed directly by the transformation operators, Fitts' Law model is analyzed afterward by calculating the movement time based on current model state's design components. This metric is used for determining whether the next target needs to be closer to the previous target to improve movement time.

Parhi et al.'s model

The model of error in selecting targets by touch due to Parhi et al. (Parhi, Karlson, & Bederson, 2006) is also analyzed when the Fitts' Law model is calculated. This model is analyzed based on the target's size as indicated in the design components model. This metric is used for determining whether to increase the target size to reduce the selection error rate. From the data points presented in graph Figure 2.4 of the relationship between the erroneous trials for each discrete target size, the following equation is used.

$$touch\ error\ rate = \boldsymbol{e}^{(\mathbf{4.7680933 - 0.092352} \times (\boldsymbol{target\ size\ in\ pixel}))}$$

I use this equation as a model for analyzing the error rate for target selection. For example, the touch error rate for the button on the first screen in Figure 4.2 is approximately 17.82 % (assuming the smaller value of width and height, 21 pixels, as the target size).

This model is analyzed after the transformation operation based on the current model state's design components, such as Fitts' Law, by calculating the movement time incorporating the error rate for selected target and indicating whether the size of the target should be increased.

Layout Complexity

Layout Complexity is a measurement of screen complexity. This measure is based on the observation that users find visually complex designs (i.e., with higher layout complexity) decrease usability (Heim, 2007). I use the formalism given by Galitz (Galitz, 2002) to calculate layout complexity. Three types of counts are summed: (1) the number of elements on the screen, (2) the number of horizontal (column) alignment points, and (3) the number of vertical (row) alignment points. For example, the layout complexity of the first screen in Figure 4.2 is 7.

This model is also analyzed after a new model state is created as a result of applying transformation operators to a prior model state. A higher layout complexity indicates that the buttons on the same screen need to be aligned or the number of the buttons needs to be reduced to reduce cost.

4.2 Model Transformation Operators

The action graph formalism is chosen as a framework for integrating other interaction models of the design; thus model transformation operators (i.e., transformation rules) are defined in terms of action graphs.

Table 4.1 Action graph operators and corresponding KLM operators with design implications

Action Graphs Operator	*Design Implication*	*KLM Operators*
add an edge	add new design element to receive user action (e.g., add a new button)	add operator sequence for added user action (e.g., add *MPK* for look for new button, point, and tap)
remove an edge	remove corresponding design element (e.g., remove a button)	remove operator sequence for the design element (e.g., remove *MPK* sequence linked to the design element)
add a vertex	add a new design element (i.e., screen) representing the state of the design	add operator sequence for added design element (e.g., add *R(t)* for system response time to load screen)
remove a vertex	remove a design element (i.e., screen) representing the state of the design	remove operator sequence for the design element (e.g., remove *R(t)* for corresponding design element)

Four primitive model transformation operators are defined for an action graph. The corresponding effects on a Keystroke Level Model as well as design implications are shown in Table 4.1. However, as my focus is on improving the efficiency of the task execution time, I will not consider 'add a vertex' operator as a primitive operator within the scope of this dissertation. Adding a vertex to an action graph requires adding edges to link the new vertex into the original model, resulting in increased task execution time.

For my model transformation approach, model transformation operators are defined to be applied on the goal path of the action graph for a task. In each state, all possible primitive operators that can be applied to a goal path are treated as model transformation operators. The goal path is defined as follows:

$$goal\ path\ of\ an\ action\ graph = v_0 \xrightarrow{e_0} \dots \xrightarrow{e_{i-1}} v_i \dots \xrightarrow{e_{n-1}} v_n$$

$$(0 \le i \le n, v_i = i^{th}\ vertex,\ e_i = i^{th}\ edge, n = length\ of\ goal\ path)$$

Model transformation operators for a state are described in the following sections. Whenever the operators are used to expand (transform) a model state, not only is the action graph transformed but also the KLM and design components in that state.

4.2.1 Add an Edge

Adding a new edge can be done by adding e_x to current goal path without adding new vertexes. However, the number of possible new edges that can be added to the goal path varies by the number of vertexes in the goal path. I consider all possible new edges that are in the same direction of the goal path (i.e., no backward links) and do not coincide with the edges already in the goal path. The path length of added edges varies from 2 to $n - 1$ (assuming the path length of an edge in the current goal path is 1). For example, if $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$ is the goal path for the model state, operators for adding an edge are $add\ edge\ e_{0,2}$, $add\ edge\ e_{1,3}$ (with path length 2), and $add\ edge\ e_{0,3}$ (with path length 3), where $e_{i,j}$ represents an edge from vertex *i* to vertex *j*.

4.2.2 Remove an Edge

Removing an edge removes one e_i from the current goal path. The number of possible operators to remove an edge is $n - 1$, excluding removing the first (or last) edge in the goal path. For example, if $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$ is the goal path for the model state, operators for removing an edge can be represented as $remove\ edge\ e_{0,1}$ and $remove\ edge\ e_{1,2}$ from the current goal path where $e_{i,j}$ represents an edge from vertex *i* to vertex *j*. However, if an edge (e.g., $e_{1,2}$) is removed from the goal path, the path is broken into two parts (e.g., $v_0 \rightarrow v_1, v_2 \rightarrow v_3$) in the new model state. To maintain the original functionality of the design, I connect these parts by adding a new edge (e.g., $e_{1,3}$) for the goal path of the new model state when this operator is executed.

4.2.3 Remove a Vertex

Removing a vertex removes one v_i from the current goal path. The number of possible operators to remove a vertex is $n - 2$, excluding the removal of the first and the last vertex of the goal path. For

example, if $v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow v_3$ is the goal path for the current model state, operators for removing a vertex can be represented as $remove\ vertex\ v_1$ and $remove\ vertex\ v_2$ from the current goal path. However, if a vertex (e.g., v_1) is removed from the goal path, the path is broken into two parts with two dangling edges (e.g., $v_0 \rightarrow , \rightarrow v_2 \rightarrow v_3$). To maintain the original functionality of the design, I connect these two by removing the dangling edges (e.g., $e_{0,1}, e_{1,2}$) and adding a new edge (e.g., $e_{0,2}$) to the goal path of new model state when this operator is applied.

Figure 4.6 shows an example of the removal of a vertex (remove Vertex 5) used for transforming the model presented in Figure 4.3. A gray rounded rectangle represents a state, and the interaction models of the design of this state are shown inside it (Fitts' Law, Parhi et al.'s error model, and layout complexity are not shown). A blue dotted line marked on the action graphs model shows the goal path. The red X in the original state (above the arrow) indicates the portion of each interaction model that will be transformed by the operator. After the operator is applied to the original model, a new state (below the arrow) is created. A red dotted circle marks the portion of the model where there were changes from to the original state. As this diagram shows, by removing a vertex from the action graph for the state, a corresponding screen is removed from the design components and the corresponding KLM operator sequence for that screen is also removed.

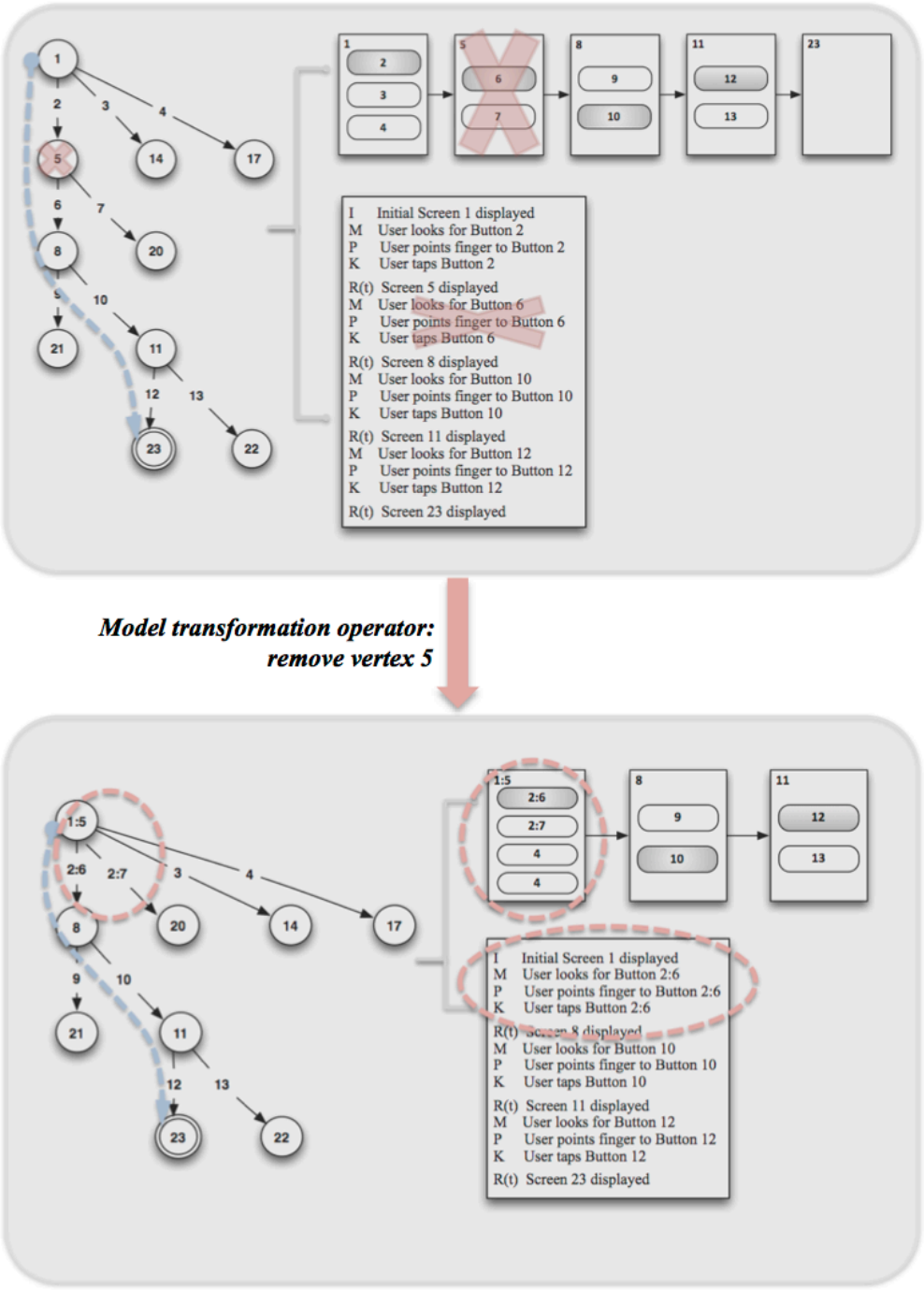


Figure 4.6 Model transformation example with 'Remove Vertex 5' operator

4.3 Implications for Design Recommendations

From the model states and operators applied in the search algorithm, the following information can be generated automatically. This information can be used to produce guidelines or recommendations to the designers for redesigning an interface to improve efficiency:

- Various usability measures based on interaction models embedded in a particular model state.
- Plausible design variations based on successor model states explored.
- Design change steps (based on transformation operator sequences) to reach a successor model state's design.
- Design change suggestions based on interaction models.

The scope of the recommendations is divided into two levels: (1) recommendations from interaction models applied during the model transformation step (i.e., action graphs, KLM, design components), and (2) recommendations based on interaction models analyzed after the model transformation step (i.e., Fitts' Law, Parhi et al.'s model of error, and layout complexity).

4.4 Discussion

In order to define a framework for the model transformation approach, I consider exploring design variations based on interaction models representing the design to be a general search problem. A model state represents the state of a design represented as a set of interaction models integrated into an action graph model. Transformation operators are defined in terms of action graphs to represent design changes that improve usability of the design of the model state. Executing the search algorithm with these components allows this framework to gather useful information to generate UI recommendations to improve usability of a given design.

Throughout this dissertation, I demonstrate the integration of four interaction models (i.e., KLM, Fitts' Law, Parhi et al.'s model of error, and the layout complexity measure) into an action graph model to represent model transformation. However, the type of interaction models for integration are not limited to the ones considered in this dissertation. The framework for the model transformation approach generalizes to other interaction models as well. In order to integrate a new model, related information for the new interaction model is either defined in terms of a design components model (i.e., as the KLM is defined) or generated in the same way as the design components model (i.e., as Fitts' Law, Parhi et al.'s model of error, and the layout complexity measure are defined). Once a transformation operator is executed with an action graph operator, the associated transformation is performed to the design components model, leading to the new design. The change in the design components model leads to the transformation to the new interaction model. Then, based on the transformation to the new interaction model, usability measures as well as design recommendations for the new interaction model are generated. For example, if a new interaction model (M) that measures visibility of the design is integrated (assuming a quantitative model based on the design components model) and an action graph operator 'remove a vertex' is executed, this transformation operator removes a screen from the design components model as in Figure 4.6. In addition to the transformation operations to other interaction models, transformation to model M results in calculating a visibility measure based on model M for the new design. Design recommendations to improve the visibility of the new design are also provided based on this measure.

Based on this model, the disadvantages of the traditional model based approach can be reduced. As the models are transformed automatically by the operators described above, the transformation approach automatically produces a new model for a new design (in the form of successor model states). The model transformation operators in theory increase user performance (e.g., reducing user

execution time); therefore the designs mapping to the new model will have better usability than the original design (i.e., a design that maps to the initial model state or parent model state, before transformation). By associating design implications with the sequence of applied transformation operator, this approach will guide the designer to improve the usability of the initial design. This should help to reduce the possibility of failure in improving the interface, and it should reduce the number of iterations required in the redesign process to improve usability.

5 A MODEL TRANSFORMATION SYSTEM

This chapter describes how the model transformation framework is implemented as a system. Information about a UI design and an interaction model are represented in a predefined format to represent model states and model transformation operators are defined to run a search algorithm. By collecting information while exploring the search space, my system produces design recommendations that can improve usability (particularly the efficiency aspect of usability) of the original design. The ultimate role for this system is to provide a tool that can be used during the UI design and development process, particularly when the goal is to improve usability of a given design.

5.1 UI representation

For the implementation, I chose the domain as a handheld touch device such as a mobile phone. The main reason for this is because as the focus of my research is on validating the effectiveness of the proposed model transformation approach, I wanted designs with less variety in design components (e.g., sets of screens with tapping areas) and limited user actions. Moreover, mobile application designs were chosen because it is simpler to incorporate information through models than with desktop designs, which include complex information in a wider variety of design components and user actions.

I selected designs from mobile application and transformed them into wireframes with screens and buttons (i.e., components with tapping areas). Also, I chose tasks that were used frequently in the applications and represented that information with usability models. For example, Figure 5.1 shows a screen design sequence users have to go through to play a new game in the Scrabble interface for the iPhone. (A green rounded box indicates a button that is to be tapped). The corresponding wireframe design for representing UI information is shown in Figure 5.2.

Figure 5.1 Scrabble: Screen design sequence for 'Play a new game' task

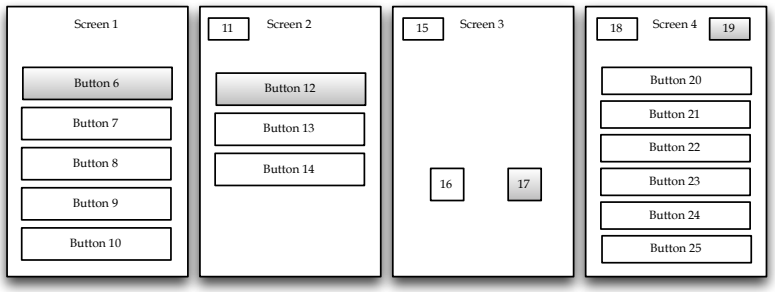


Figure 5.2 Scrabble: Wireframe design sequence for 'Play a new game' task

For each mobile application UI design, I defined three kinds of model information to run the system as described in Chapter 4. For this implementation, to reduce the size of the search space and to focus on validating the effectiveness of the model transformation approach, I used a wireframe design created based on an actual application UI design with buttons and screens as design components. I considered any contiguous area that can be tapped as a button. In addition, I made a few more assumptions. All buttons are associated with one event that causes a transition to a different screen, a transition to a different state of the device. I associated same KLM operator for that event. Although I reduced the scope of the designs for the analysis, the formats and algorithm of the system don't limit its scope to only simple designs. It is implemented to accommodate other information as well, including other design components, various user actions, and a range of design or event constraints.

5.1.1 Design Components Model

Design information is represented based on the wireframe design of the UI. For representing a Design Components Model, I selected information appropriate for the use of usability models. I defined the following information fields for a Design Components Model per each design component of the UI.

- Type
- ID
- Starting Position (X, Y)
- Size (Width, Height)
- Event (Type of gesture/action, Current Screen ID, Next Screen Id)
- Constraints

A *Type* field represents the type of design component. For example, 'S' is marked for a screen component and 'B' is marked for a button design component. An *ID* field is a numerical id of this design component (i.e., Screen ID, Button ID) that is unique among the design components of the UI. A *Start position* field represents the upper-left corner position of the component area in pixels, with the upper-left corner of the screen as the starting point (0, 0). A *Size* field represents the width and height of the size of this component in pixels. An *Event* field represents an application event tied to this component. If a specified gesture (e.g., tap, drag, swipe, etc.) is performed on this component in the current screen (i.e., current screen's design component ID) it cause a screen transition to the next screen (i.e., the next screen's design component ID). A *Constraints* field specifies the design constraints on this component. For example, the size of the component can be marked as controllable /uncontrollable or the position of the component as fixed/unfixed. Any design constraint that will influence design recommendations (i.e., system output) is specified in this field.

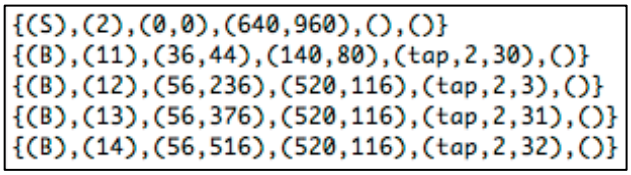


Figure 5.3 Scrabble: Design Components Model representation of Screen 2

A Design Components Model for the second screen (Screen 2) in Figure 5.2 is shown in Figure 5.3. The first line in this model represents the information for the screen itself. The other lines show information about the buttons in this screen. The buttons' association with a screen is specified in the *Event* field (5th field) with the Screen ID. For instance, (tap, 2, 3) in third line indicates that Button 12 is located on Screen 2 and that tapping this button results in a transition to Screen 3.

5.1.2 Action Graphs

An action graph is represented with states and actions that cause transitions in the application. I represent a state as the state of the device (e.g., screen design) and an action as the user action performed on certain design component (e.g., button). I defined the following information fields for each transition for the action graph of a design.

- Current State (Screen ID, Button ID, Type of user action)
- Next State (Screen ID, Button ID, Type of user action)

Three information fields represent an action graph state. *Screen ID* (i.e., the design component ID of a screen) represents the ID of the current state. *Button ID* (i.e., a design component ID of a button) and *Type of user action* (i.e., type user action performed on the button) fields represent an event that causes a transition to the current Screen (i.e., performing a user action to the button causes a state

transition to the current state). With this state information, a transition is represented by specifying the current state and the following state, presented with an arrow sign. As the model transformation operators are applied on the goal path (i.e., shortest path on action graphs from initial screen to frequently accessed screen) in an action graphs, I mark the start state (i.e., initial screen) and a goal state (i.e., goal screen)these information in this the representation.

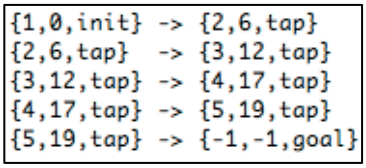


Figure 5.4 Scrabble: Action graph representation of design sequence

The action graphs for the screen design sequence and task sequence in Figure 5.2 is shown in Figure 5.4. A state on the left side of the arrow of the first transition is the initial state *{1,0,init}* representing Screen 1. The last transition indicates that *{5, 19, tap}* is the goal state (Screen 5 is assumed as a screen for playing the actual game). The first transition represents the first screen transition of the sequence, Screen 1 to Screen 2. Performing a tap gesture on Button 6 in Screen 1 causes this transition. The second transition denotes that performing a tap gesture on Button 12 in Screen 2 causes a transition from the previous state (Screen 2) to Screen 3. Note that transitions should be defined for all possible events (i.e., user actions) allowed in the design (exclude in Figure 5.4.). There can be several transitions starting from the same state; however, all transitions specified for an application are unique.

5.1.3 Keystroke Level Model

The Keystroke Level Model is represented with information about KLM operators as well as corresponding design component information. I define the following information fields per each operator for the KLM model of the design.

- KLM operator
- Subject of the operator
- Type of action
- Button ID
- Screen ID

KLM operator represents the KLM operators (e.g., *K*, *P*, *M*, *I*). As my designs are for touch-based handheld mobile device, I use the K operator to represent tap gestures/actions and the P operator to represent finger movements. *Subject of the operator* represents whether the operator is the user's action or the system's operation. This field is specified as *USR* for user actions and *SYS* for system operations. *Type of action* specifies the task that corresponds to the operator. For example, *look_for* is for the task of visually searching for a target, *point_to* is for the task of moving finger to a target, *tap_to* is for the task of tapping a target, and *display_screen* denotes a system operation of displaying a screen. *Button ID* is a design component ID of the target; however, I use screen ID in this field when the target is an entire screen. *Screen ID* is a design component ID of a screen where this operator was analyzed.

```
{(M),(USR),(look_for),(12),(2)}
{(P),(USR),(point_to),(12),(2)}
{(K),(USR),(tap_to),(12),(2)}
{(R),(SYS),(display_screen),(3),(3)}
```

Figure 5.5 Scrabble: KLM representation of 'Tap Button 12 and transition to Screen 3'

The KLM for the task sequence of 'Look for Button 12, Point to Button 12, Tap Button 12, and Display Screen 3 (i.e., look for the Local Play button on Screen 2 and tap it to proceed to Screen 3)' is shown in Figure 5.5. As the KLM operator sequence of this task is 'MPKR', each line in the model represents each operator sequence with the corresponding information. The first line is the operator is M; when the user performs this task, the kind of task is 'Look for a target', the target of the task is Button 12, and the task is performed on Screen 2.

5.2 Model Transformation

When designs and tasks are represented in the form of models (as described in section 5.1), the system can analyze the usability for the selected application design and present information that can be useful on making decisions to improve the design. Assuming that the user understands the design, the initial screen and the goal screen of the design, a usage scenario for the model transformation system is as follows.

(The system lists the name of the application.)

1. Select the name of the application.

   (The system presents usability analysis measures for the original design.)

   (The system presents a list of design variations based on possible screen sequences.)

2. Select a design variation.

   (The system presents usability measures of the selected design variation.)

   (The system presents one possible design change step to achieve the selected design variation.)

   (The system presents additional design change recommendations for further improvement.)

3. Apply the suggested recommendations for design changes to improve usability, or select another design variation.

The system starts the analysis when the name of the application (models for the application must be represented and imported before starting the system) is selected from the presented list (Figure 5.6).

Figure 5.6 List of the applications to analyze

5.2.1 Analysis of Original Design

For the selected application, the system shows four kinds of information to the user: (1) information about the goal path, (2) usability measures for the original design given the goal path, (3) the layout of design components used for computing usability measures, and (4) a list of design variations with a different set of screen sequences.

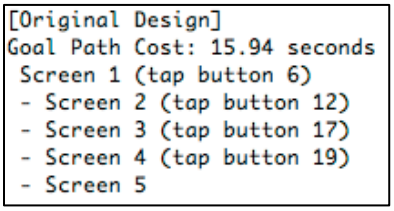


Figure 5.7 Screen sequence and cost of shortest goal path

Figure 5.7 shows the first presentation the system gives, for the screen and the user action sequence corresponding to the goal path in action graphs of this design (i.e., a path with the shortest user execution time) found to reach the goal screen for the design in Figure 5.2. In addition to seeing the screen sequence of the goal path, the user can see the total KLM execution time for the task (i.e., Goal Path Cost).

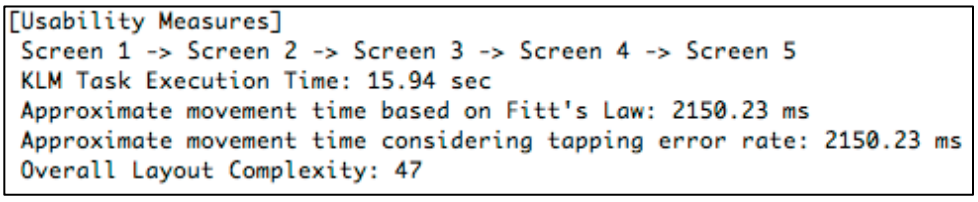


Figure 5.8 Usability costs based on various measures for original design

The system then computes the usability cost of the task, based on the screen sequence of goal path, with usability models embedded in the system (i.e., Fitts' Law, Parhi et al.'s model of error in selecting targets by touch, layout complexity measures). These models provide estimates that correlate with KLM duration but differ based on modeling assumptions and level of detail. Figure 5.8 shows this information with the screen sequence to reach the goal screen. Task execution time using KLM model is computed by adding up the operator time of corresponding KLM operator sequence of the goal task. Approximate movement times are computed based on the design components' positions arranged automatically by the Java FlowLayout Manager (Oracle, 2014). Although we know the position of each design component for the original design, the proper position of the components when there is a change in the design (e.g., adding a new button or removing an existing button) may be ambiguous. We need a consistent scheme for arranging buttons on the screen. Thus in current system, I create a JPanel Object for each screen and use the FlowLayout Manager to arrange the buttons on that screen. The positions of the automatically arranged buttons are used to calculate approximate movement times. (Because the FlowLayout Manager arranges buttons left-to-right then top-to-bottom and center-aligned based on their size, the automatically generated screen design does not always quite match actual design; this should be viewed as a modeling approximation.) Figure 5.9 shows how buttons are arranged on the screen for the design in Figure 5.2 when the FlowLayout Manger is used. Approximate finger movement times given a model-based tapping error rate and layout complexity is also calculated for the design and presented to the user.

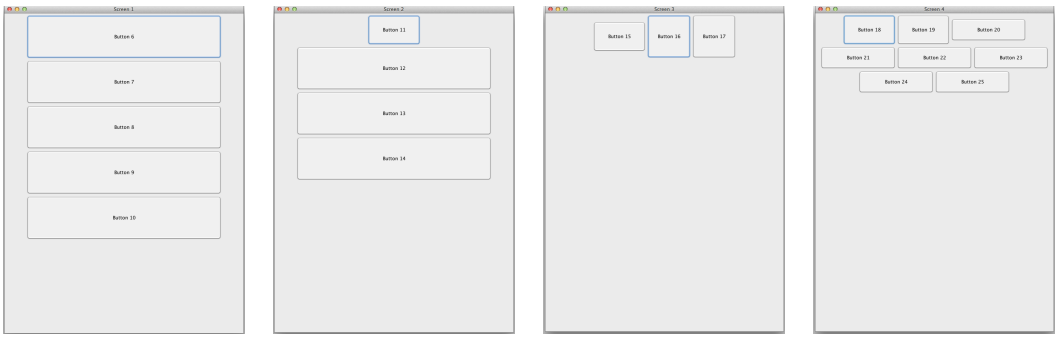


Figure 5.9 Design components of original design arranged with Java FlowLayout

5.2.2 List of Design Variations

Lastly the system presents a list of design variations, composed of a set of permutations of screen sequences to reach the goal screen. I made an assumption for each screen sequence in the design variations: the length of the screen sequences should be equal to or shorter than the original sequence to reach the goal screen. This assumption is based on the fact that as the number of screens increases to reach the goal screen, the task execution time increases proportionally. An increase in task execution time is generally considered a degradation to usability. This list is also a unique set of screen sequences of successor model states produced by transformation operators. When the design variation list is shown, the user can select one specific design variation based on the screen sequence presented. The user will then be able to browse the design implications that represent design recommendations to achieve the screen sequence of the selected design variation. Figure 5.10 shows a list of design variations for the design in Figure 5.2

```
[Possible design candidates]==============================================
System considered 24 kinds of design variations
[Design   #] Task time : task sequence
[1] Screen 1 -> Screen 5
[2] Screen 1 -> Screen 2 -> Screen 5
[3] Screen 1 -> Screen 3 -> Screen 5
[4] Screen 1 -> Screen 4 -> Screen 5
[5] Screen 1 -> Screen 2 -> Screen 3 -> Screen 5
[6] Screen 1 -> Screen 2 -> Screen 4 -> Screen 5
[7] Screen 1 -> Screen 3 -> Screen 4 -> Screen 5
[8] Screen 1 -> Screen 2 -> Screen 3 -> Screen 4 -> Screen 5
==========================================================================
```

Figure 5.10 List of design variations by difference screen sequences

5.3 Implications for Design Recommendations

When the user selects a design variation, the system provides design implications to the designer. Most of the information in this step is collected during the model transformation step. The purpose of providing such information is not to give the user a single correct recommendation but instead to provide relevant information for their design decisions. The quality of recommendations will depend on the model embedded in the system. The system will help validate designs in the context of those models. The designer can consider other aspects of usability that might be missed by the models to make the final design decision.

When the designer chooses a design variation, the system provides four kinds of information: (1) usability measures of the new design with the improved path to the goal (i.e., the screen sequence of the selected design variation), (2) the layout of design components used for computing usability measures, (3) design recommendations for changing a screen sequence to produce the selected goal path, and (4) further design recommendations to further improve the selected goal path.

5.3.1 Analysis of the Selected Design Variation

```
[Detailed design improvement Information of Design 2]--------------------
 Screen 1 -> Screen 2 -> Screen 5

[Usability Measures of selected design]
 [46.30 % improved] KLM Task Execution Time: 8.560 sec
 [35.46 % improved] Approximate movement time based on Fitt's Law: 1219.580 ms
 [35.46 % improved] Approximate movement time considering tapping error rate: 1219.580 ms
 [0.00 % improved] Overall Layout Complexity: 26
```

Figure 5.11 Usability cost based on various measures for selected design variation

For a selected design variation, an overall usability analysis and the design recommendation related to it is provided first. Usability measures are based on the same usability models used for analyzing original design. **Figure 5.11** shows this information when design variation choice 2 is selected from the design variations listed at Figure 5.10. To analyze movement time, I use the FlowLayout Manager to arrange the buttons of the new design, as in the analysis of the original design. This layout (Figure 5.12) is used for calculating movement times and layout complexity of the selected goal path.

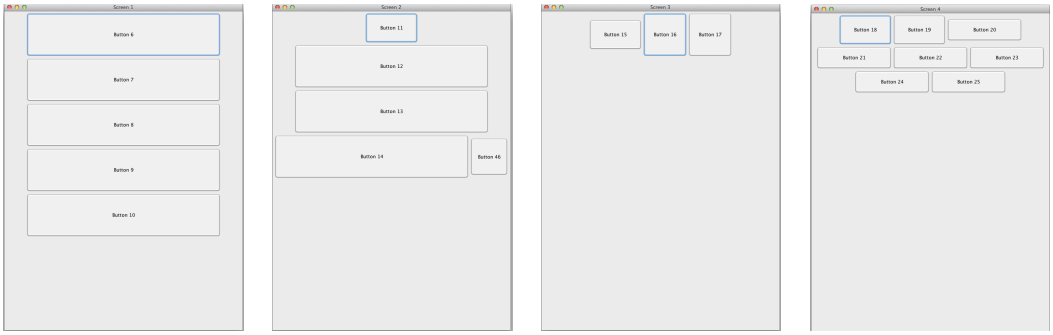


Figure 5.12 Design components of selected design arranged with Java FlowLayout

5.3.2 Basic Recommendations to Change the Goal Task Path

A basic recommendation presents one possible step to change the original design to the new design matching selected design variation. This recommendation is generated based on the order of the model transformation operators applied in model transformation algorithm to reach the model state of selected design variation. There can be several possible operator sequences to reach the same model state. However, my current system is implemented to present the recommendation with shortest model transformation operator sequence. Figure 5.13 shows basic recommendation for design variation choice 2 from list at Figure 5.10.

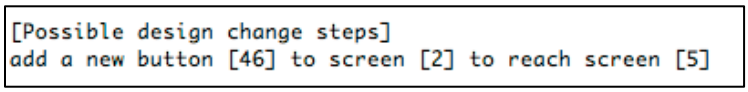


Figure 5.13 Design recommendation to satisfy selected goal path

5.3.3 Further Recommendations to Improve the Goal Task Path

Further recommendations are provided through analyzing the usability cost based on interaction models. Three models (Fitts' Law, Parhi et al.'s model of error, and the layout complexity measure) are used for generating these recommendations. In order to improve the Fitts' Law movement time, the system suggests moving the button in the goal path closer to the position of the previously tapped button. The system also recommends increasing the size of a button based on Parhi et al.'s model if the touch error rate is beyond a fixed threshold (currently set to 4% (Wobbrock, Cutrell, Harada, & MacKenzie, 2008)). Moreover, it suggests aligning buttons on a screen if the layout complexity of that screen is greater than a defined threshold set (currently set to (*number of buttons on a screen* × 2

+ 1) to minimize the alignment points (Galitz, 2002) by assuming all buttons in a screen is aligned). Figure 5.14 shows these further recommendations design variation choice 2 from list at Figure 5.10.

```
[Additional design recommendations]
  - Consider repositioning following button's closer to indicated position.
   Move Button 46 closer to Button 6 's position
  - Consider aligning button's on the following screens.
    Screen 1
    Screen 2
```

Figure 5.14 Design recommendation to improve selected goal path

5.4 Search Space Analysis

The benefit of using this model transformation system is that the designer can go through information relevant to improving the usability of original design more effectively than relying only on the designers for searching and collecting relevant information for the improvement. Because this information is collected automatically with the model transformation approach, it is not only correct in the context of improving efficiency but also well-supported with corresponding interaction models that quantitatively measures usability. In addition, the system automatically generates possible new design candidates and analyzes their usability within the scope of the specific models used, so that the designer can rely on the tool for exploring the space of possible design candidates and their usability cost.

In this section, I analyze the size of search space of possible design candidates covered by the tool and analyze the outcomes of my model transformation system.

5.4.1 Quantitative Analysis

My system focuses on improving the efficiency of the goal task path (i.e., the screen and button sequence to reach the goal screen; the shortest path from initial state to the goal state in action graph). Thus I analyzed the number of model states in the space searched by the model transformation approach based on the length (i.e., the number of buttons tapped) of the goal path. For each state in the search space, the number of operators are defined based on the goal path length of action graph for that state. For example, if the number of buttons tapped for the goal path is *n*, the number of operators in the *add an edge* category is $\frac{n(n-1)}{2}$, the number of operators in the *remove an edge* category is $n-1$, and the number of operators in *remove a vertex* category is $n-1$. Therefore the total number of operators considered in the first search expansion is $\frac{(n+4)(n-1)}{2}$. When these operators are applied to the initial state, the interaction models in the state are transformed to the successor states; the design components in the goal path changes accordingly. Based on the changed goal path, transformation operators are again determined for each successor model states to expand (transform). The number of operators increases to $\sum_{i=1}^{n-1} i^2 + 2(n-1)^2$. The model transformation algorithm does not expand duplicate states (i.e., it checks whether an expanded state is in the current *open list* or *closed list*), to reduce the number of states considered in the search.

Table 5.1 Analysis of the number of states and operators

Length of goal path of initial state	*Number of operators of initial state*	*Total number of states explored*	*Time spent on exploring search space (ms)*	*Number of unique goal paths of successor states*
2	3	4	13	2
3	7	26	90	4
4	12	244	691	8
5	18	3016	1930	16
6	25	46304	15681	32
7	33	849616	683237	64

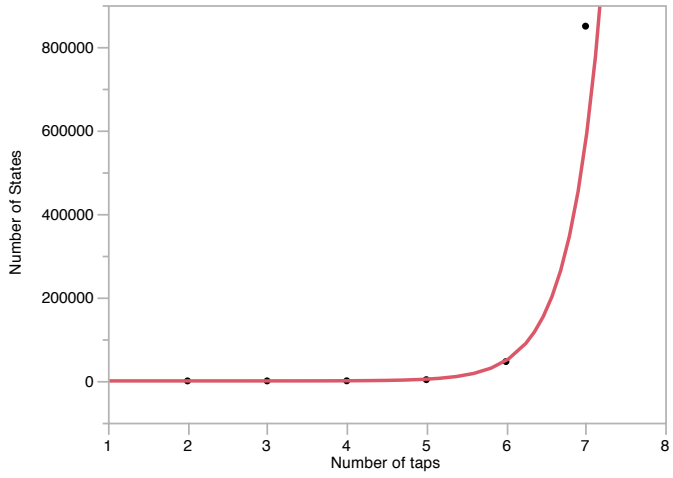


Figure 5.15 Number of taps versus number of states

Table 5.1 shows the number of operators applied to the initial state (with the original design and goal path), the number of states explored by the model transformation algorithm, and the number of unique goal path over the set of new design candidates (i.e., design variations) per the length of the goal path of initial model state. The number of unique goal paths of explored successor states increases with 2^{n-1}, assuming *n* as the length of the goal path of the initial state.

Figure 5.15 shows a graph of the total number of states explored in the search space by the length of the goal path (i.e., the number of taps required to reach the goal screen) of the initial model state. These data points can be fit by an equation, shown as a red line in the graph:

$$\log(number\ of\ states) = -4.004305 + 2.4657259 \times number\ of\ taps$$

As expected, the number of states in the search space explored by the model transformation system increases exponentially with the number of taps required to reach the goal screen (i.e., the length of goal path of initial model state). The actual computational time spent in exploring the search space (on an Apple MacBook Air with OSX 10.9.4, 1.7 GHz Inter Core i5 processor, 4GB 1.3 MHz memory) is shown in the fourth column. The increase is also exponential.

5.4.2 Qualitative Analysis

The model transformation system presents a variety of information to the designer. The designer may adopt the provided information for improving the usability of a given design. In this section, I describe how the designer can interpret the provided information for this purpose.

Usability cost of Original Design

The system first finds a shortest task sequence (i.e., a sequence of buttons to tap on a screen) to reach the goal screen. Various interaction models are used to analyze efficiency of the original design. Usability cost is calculated as task execution time using the KLM, movement time based on Fitts' Law, finger movement times given a model-based tapping error rate, and overall layout complexity. If the designer judges by any of these metrics that the current design is less than satisfactory (e.g., that the total execution time should be less than *x* seconds, or that the tapping/pointing error rate should be less than y%), then the designer can improve the design as necessary. This is the same kind of

information that the designers rely on when they follow a traditional model-based approach to improve a design. Whereas it is totally up to the designers to make proper redesign decisions when following the traditional approach, my system provides guidance for creating new designs that guarantee improvements in usability.

List of Design Variations

If a design change is needed, the designer can first go through a list of design variations as represented by different screen sequences to reach the goal screen. Because the context or the semantics of the design is not represented in the models, some of automatically generated screen sequences of design variation may not be appropriate for the application. However, providing this level of information is still reasonable, so that the designer can take efficiency information into account while considering design changes that are sensitive to context. I also expect the information systematically generated by the system to aid the designer in considering changes that might otherwise be missed.

Usability cost of Selected Design Variation

The list of design variations gives the designer a rough idea about possible changes to the sequence of screens and the length of goal path (with corresponding changes in task execution time). When a certain design variation is selected, the usability costs based on task execution time using the KLM, movement time based on Fitts' Law, finger movement times given a model-based tapping error rate, and overall layout complexity are also calculated and shown. In addition to these costs, the difference as a ratio between each cost metric in the new design and in the original design is provided. The designer can then check whether selected design variation is above some threshold.

Basic Recommendations

The system recommends a way to change the original design to the selected design variation. For instance, if the original design's screen sequence is Screen 1 → Screen 2→ Screen 3 and the selected design variation is Screen 1→ Screen 3, there are several ways to change the original design to the new one. By tracing the model transformation operators, 'Add a new button on Screen 1 that reaches Screen 3', 'Remove Screen 2 and add a new button on Screen 1 to reach Screen 3', or 'Remove a button on Screen 1 that reaches Screen 2 and add a new button on Screen 1 to reach Screen 3' are possible redesign steps. My system provides one of these changes to help designers in the process of creating a new design with better usability, while maintaining the same functionality of the original design.

Further Recommendations

Based on usability costs measured for the selected design variation, more design recommendations are provided based on each interaction model result: (1) Move a button closer to the previously tapped button, to improve movement time based on Fitts' Law, (2) Increase the size of a button, for those with a larger pointing/touching error rate than a threshold, to decrease the error rate (and thus to improve movement time as computed taking the error rate into account), and (3) Align buttons or reduce the number of buttons to reduce the complexity of the screen.

5.5 Discussion

In order to implement a framework that simulates the model transformation approach, I define three components necessary for the system: (1) a UI representation module, (2) a model transformation module, and (3) a design implications module. This reflects the conceptual flow shown in Figure 4.1.

For the UI representation module, the Design Components Model is defined to incorporate the interaction models used in the system. The interaction models can be divided into two categories: models that must be defined in addition to the Design Components Model during the model transformation step (e.g., action graphs, KLM), and models that can be analyzed after the model transformation step (e.g., Fitts' Law, Parhi's model of error, layout complexity measures). In the model transformation module, a search algorithm is implemented with states in the form of models as defined in the UI representation module. This algorithm performs model transformation operations and collects information produced in the exploration of the search space. In the design implication module, recommendations are generated based on the information collected by the model transformation module.

This demonstrates that the model transformation approach can be realized with the implemented system. The development also brought to light issues at the implementation level. For instance, resolving the layout issue in design is necessary for generating design recommendations, specifically recommendations that are based on the interaction models.

Although the model transformation approach is useful, it has limitations. First, if a designer has enough knowledge and experience in designing UIs as well as analyzing usability through modeling, then my system may not be helpful. However, as the analysis on the size of search space shows (that the search space grows exponentially with the number of user actions for the goal task), it will be hard for the designers to go through all the design alternatives by themselves; subtle possible changes may be missed. Second, the performance of the recommendations will depend on the type of interaction models chosen for the system and the level of information specified in the Design Components Model. As each interaction model presents a different performance aspect of the UIs, interaction models to embed in the system should be chosen with respect to the goal of the redesign (e.g., to

improve user execution time, to reduce complexity of design, etc.) Third, some of the interaction models such as the ones specifying a user task (e.g., KLM) are difficult if not impossible to generate entirely automatically. These models must be constructed either by hand or with the help of a tool in a separate file. This can result in more sophisticated recommendations; however, it requires preliminary effort (i.e., constructing the model) before running the system. Fourth, some issues need to be investigated in depth to increase the performance of the recommendations. Detailed representations of constraints on the design, including conflict checking and constraint prioritizing as well as proper layout, will contribute to the quality (e.g., consistency with the original design) of the recommendations as well as of the UIs. Fifth, I currently assume that the general domain of my approach, mobile interface designs, can be extended to interfaces in other domains (e.g., desktop computer applications, smart TV applications, web applications). Investigation of this issue remains as future work.

6 EVALUATION

The core question addressed by this research is whether the model transformation approach can help designers to produce user interfaces with better usability than the traditional approach. This chapter describes experiments conducted to validate the effectiveness of the model transformation approach using the system I have implemented. Experiments and analyses are designed to answer following two research questions and one exploratory question.

- Do participants use the recommendations provided by the model transformation approach to improve the usability of the designs they are given?
- Does following a recommendation actually improve an interface design?
- Does the model transformation approach contribute to design rationale?

This chapter is organized in two parts. The first part, Section 6.1, describes a two-stage experiment: the first stage studies how participants improve wire-frame models of user interfaces in two conditions (with and without UI recommendations provided), which is conventional practice in HCI; the second stage analyzes design rationale between different versions of UIs. The second part of this chapter, Section 6.2, analyzes the data from the two studies to show the effectiveness of the model transformation approach.

6.1 Experiment

The goal of the experiment described in this section is to determine whether my approach is useful for improving usability in the interface design and development process. The experiment explored the potential benefits of the model transformation approach in two ways. The first part of the experiment, which I will refer to as the *Design Improvement* component, was designed to evaluate the

effectiveness of my approach to model-based user interface development. The second part was designed to determine whether the model transformation approach can be used to predict the design rationale behind the decisions to make design changes; I will call this the *Rationale* component. One set of participants was used for both parts.

6.1.1 Participants

For this experiment, I needed participants who could act as a UI designer or developer. Specifically, I was interested in recruiting participants who have knowledge of Human Computer Interaction as well as designing and developing user interfaces. I recruited graduate students in a Human Computer Interaction class to participate in the experiment. The course was a conventional survey of topics in HCI, using *The Resonant Interface* (Heim, 2007) as the textbook. The experiment was conducted at the end of the semester, and at that point students had an overall understanding of general topics in HCI, including the interaction design process, design principles, and interaction models. Also, they had completed assignments on building and analyzing the usability of interfaces using interaction models. Therefore, students were not only familiar with usability aspects of UI design but also had experience in analyzing UIs using performance models by the time that they participated in the experiment.

Students were given an assignment to go through the tasks of the experiment, which contained online instrumentation (a survey and submission facilities). They received credit for participating, the value of a single assignment, about 5% of their course grade. (A full grade was given to students who went though the experiment to the end, but partial credit was given to students who completed only part of the experiment.) In the analysis of experimental results, each participant was identified by a randomly assigned unique number (ID). Participants' data was stored and managed using those IDs.

6.1.2 Materials

Participants provided demographics and completed the tasks described. These tasks were designed to assess the effect and potential usefulness of the model transformation approach. Participants had to go through each task, dealing with actual mobile user interface designs and with tradeoffs that conventional HCI practitioners consider when improving the usability of an interface design.

6.1.2.1 Demographic Survey

In addition to general demographic information (e.g., age, gender), the survey contained questions about participants' experience in developing and designing user interfaces, experience in the use of mobile devices, and the interaction models they had knowledge of.

6.1.2.2 Design Improvement

The Design Improvement component of the experiment was designed to assess the effectiveness of my approach compared to the conventional model-based user interface development process. It focused on improving usability, particularly the efficiency of user interaction on user interface designs for mobile devices.

Participants solved two sets of design problems on three actual mobile application interface designs. For purposes of later discussion, I will refer to the first set of problems as having been carried out in the *Traditional Condition*, the second set of tasks in the *Transformation Condition*. The conditions were presented sequentially, Traditional and then Transformation, with all participants seeing both. In the first condition, participants followed a traditional model-based usability evaluation process to improve a set of user interfaces; in the second they used the UI recommendations produced by my system to further improve their results. This sequential presentation of Traditional then

Transformation was to examine whether the model transformation approach adds value to the traditional approach.

Traditional Condition

The purpose this condition was to investigate how HCI practitioners could improve usability of interface design, by following a traditional model-based usability evaluation process. The Traditional Condition supports usability analysis based on a single common interaction model, Fitts' Law, to motivate design change decisions.

The following is the general set of instructions for this condition, shown before the participant started each design problem.

> *In the following problems, three mobile application designs will be presented. Each design shows a sequence of screens that the user goes through to complete a certain task. A task description for each screen and total task execution time calculated based on a Fitts' Law model will be presented as well.*

After these instructions, three problems were presented. In each problem, a screen shot sequence of a mobile application with a corresponding description of a user task was presented. In addition, movement time was provided, based on Fitts' Law, for the given design and user task. The following more specific instructions and questions were provided for each problem.

The design shown below is from the actual mobile application name of the application. It shows a sequence to perform a specific task. Here, the task is to short task description. Each screen size is 640 pixels wide by 1136 high. The target area user will tap to move to the next screen is indicated by an orange rounded box. A task description and usability measure based on Fitts' Law is also provided.

1. Create a wire-frame design using a prototyping tool of your choice (e.g., power point, omnigraffle, paint).

2. Change the design to improve the efficiency of the given task.

3. Describe your design improvements in detail. Support your design change decisions with a discussion of how each change improves the efficiency of design.

4. Save your improved wire-frame design and your description of the improvements in a single file and submit.

To help the participant understand the context, I provided an example problem before the participant started to solve the actual problems. The example gave a sample answer or explanation for each of the bullet points in the instructions. For instance, I added 'You can move around the buttons, resize the buttons, rearrange the buttons, change their shape, etc. You may come up with any ideas you like to improve the given design.' for the second instruction step. One example answer for the third instruction step was 'I switched the position of the Add to Cart button and the Buy Now button on Screen 6 to reduce finger movement time between Screen 5 and Screen 6.' The names of the screen and button were described in terms of the example wire-frame design given. These sample answers were intended to show the participants appropriate decisions about design changes (those that improve the efficiency of the design) making reference to the single usability evaluation model provided for the problem.

When the participant completed each problem, he or she submitted a document containing a wire-frame screen design sequence matching the original UI, a wire-frame sequence for the improved UI, and a description of the design change decisions.

Figure 6.1, Figure 6.2 and Figure 6.3 show the problems given to the participants. The designs for each problem are from the actual screen shots of the following mobile applications: the Facebook[6] application (Figure 6.1), the TransLoc[7] application (Figure 6.2), and the Zipcar[8] application (Figure 6.3). Each problem shows a corresponding user task description for the design and the total movement time as calculated by Fitts' Law. Three sets of designs were presented in random order.

Figure 6.1 Facebook: Close the chat window and hide the chat icon

[6] Facebook is an application used for social networking, developed and distributed by Facebook, Inc. Screen shots used in Figure 6.1 was captured from a mobile version (iPhone 5) of the application in 2014.
[7] TransLoc is an application used for tracking the bus route, developed and distributed by TransLoc Inc. Screen shots used in Figure 6.2 was captured from a mobile version (iPhone 5) of the application in 2014.
[8] Zipcar is an application used for finding and reserving Zipcar, developed and distributed by Zipcar. Screen shots used in Figure 6.3 was captured from a mobile version (iPhone 5) of the application in 2014

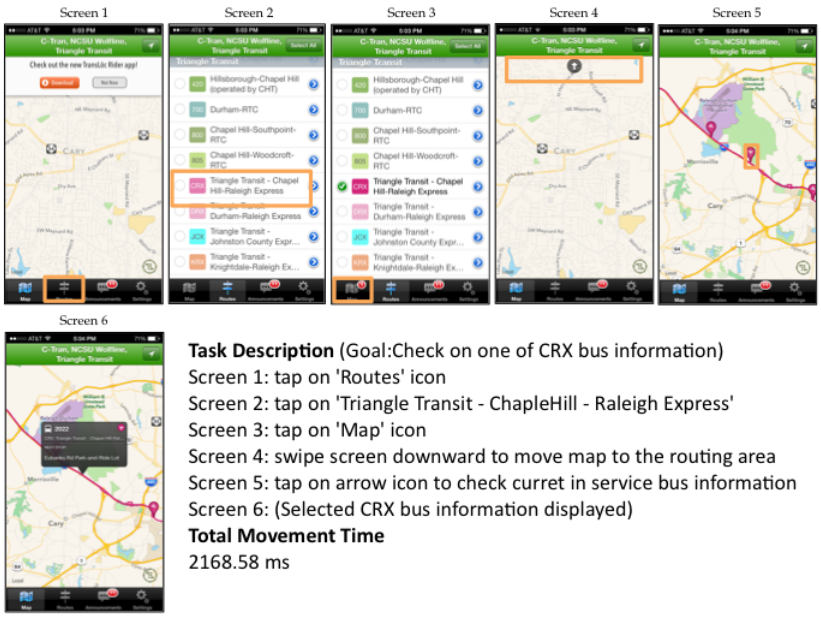


Figure 6.2 TransLoc: Show one of the CRX bus information items

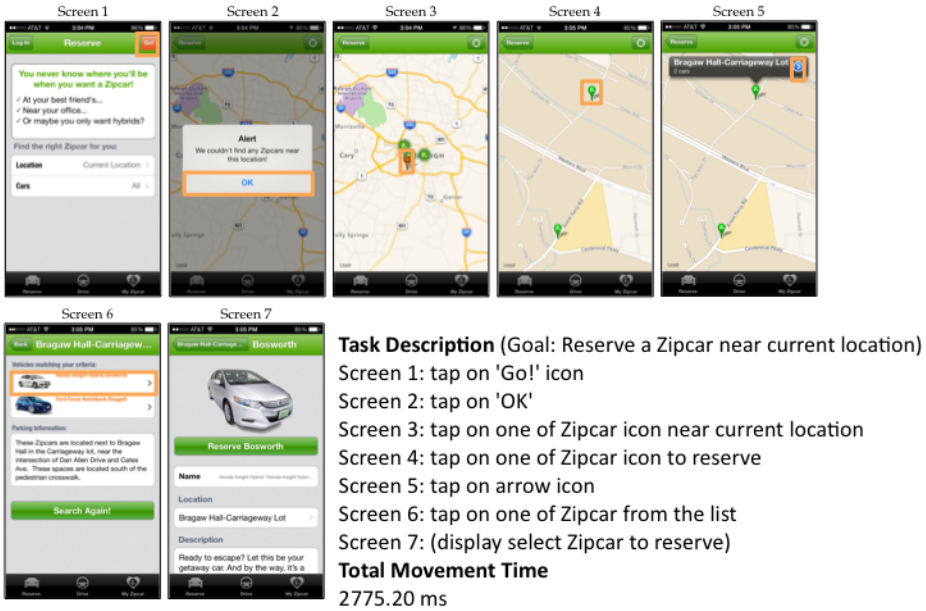


Figure 6.3. Zipcar: Reserve a Zipcar near the current location

Transformation Condition

The purpose of this condition was to investigate how HCI practitioners can improve the usability of an interface design using a model transformation approach. The Transformation Condition showed participants the results of an automated analysis for each design. Specifically, the analysis was based on multiple interaction models for measuring usability potentially relevant to their design change decisions. The possible suggestions for improving the design were presented as different screen sequences, from the starting screen to the final screen (i.e., a path to the user's goal for the task).

These were the general instructions for this condition.

> *In the following problems, wire-frame design of mobile applications from the earlier problem will be presented with the design. In addition, more usability efficiency measures as well as improvement suggestions based on the wire-frame design will be shown.*

After these instructions, three sets of problems were presented, using the same interface designs presented in the same ordering as in the Traditional Condition. In each problem, one possible wire-frame design sequence was shown. In addition to the information given in Traditional Condition (i.e., a description of the user task for the UI and the task execution time based on Fitts' Law), more usability measurements were given (i.e., task execution time as calculated from the Keystroke Level Model, approximate finger movement times given a model-based tapping error rate, and an overall layout complexity score). Below are the instructions and questions for each problem.

In the following problems, wire-frame design of mobile applications from the earlier problem will be presented with the design. In addition, more usability efficiency measures as well as improvement suggestions based on the wire-frame design will be shown.

1. Open your wire-frame design from the previous problems.

2. Select a design variation that matches with the design sequence steps, or select a design variation you'd like to use to further improve your design.

3. If appropriate, improve your previous design using the information you have selected.

4. Describe design changes in detail. (a) Specify which screen sequence information you selected. (b) Specify which information you applied or that helped with your redesign decision.

5. Save your improved wire-frame design and your description of the improvements in a single file and submit.

There are several differences from the Traditional Condition. First, the wire-frame design sequence was shown in addition to actual screen shot sequences of the application and the task description. Second, more usability analysis results were provided, based on the KLM, Parhi et al.'s model of error, and the layout complexity model (Figure 6.4, Figure 6.5 and Figure 6.6). Third, a list of design variations and corresponding design change recommendations was provided (Figure 6.7). Each problem asked participants to indicate a design variation to improve the designs they produced in the *Traditional* Condition. The design variation list was composed of a set of permutations of screen sequences to reach the goal screen. Participants could browse through variations and design change recommendations to improve their designs. For instance, the Facebook task is shown in Figure 6.4, with the screen sequence Screen 1 → Screen 2 → Screen 3 → Screen 4. A design variation is shown in Figure 6.7, in which an alternate sequence of screens is suggested: Screen 1 → Screen 2 → Screen 4. If the participant chose this variation, a pop-up window gave relevant usability information. First, the participant saw usability measures of the efficiency of the selected design variation, for comparison with the original design. Third, the participant received guidance on how to change the original design to reach the selected design; in this case the participant can do this by adding a shortcut button. Fourth, additional design changes were shown that participants could choose to make further improvements.

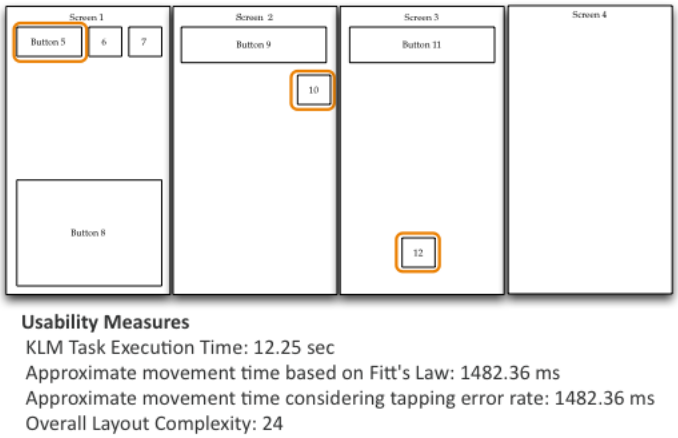


Figure 6.4 Facebook: Close the chat window and hide the chat icon

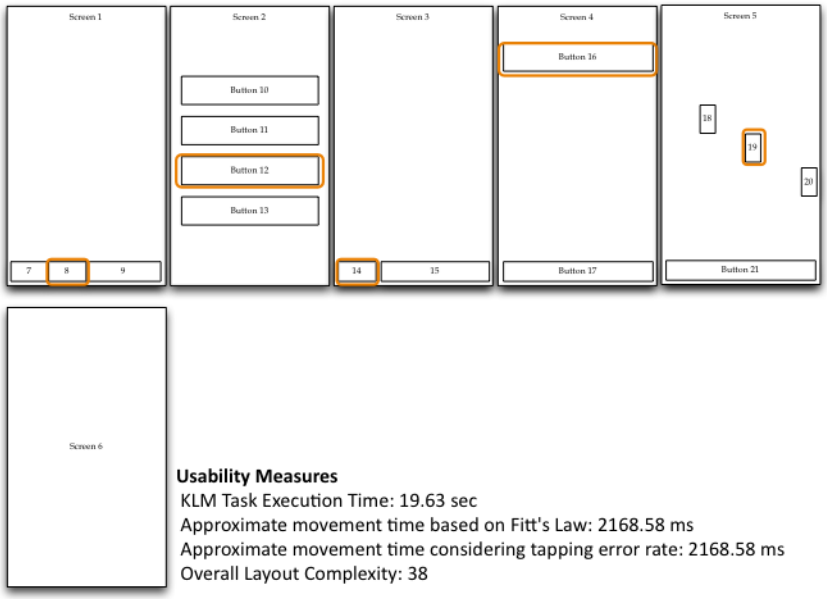


Figure 6.5 TransLoc: Show one of the CRX bus information items

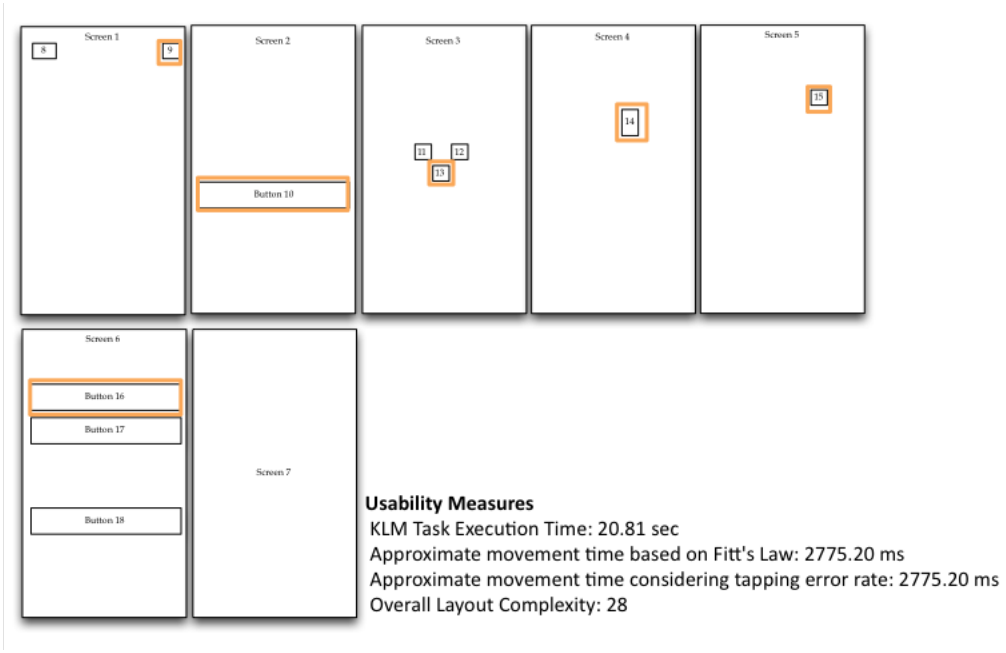
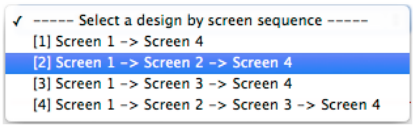
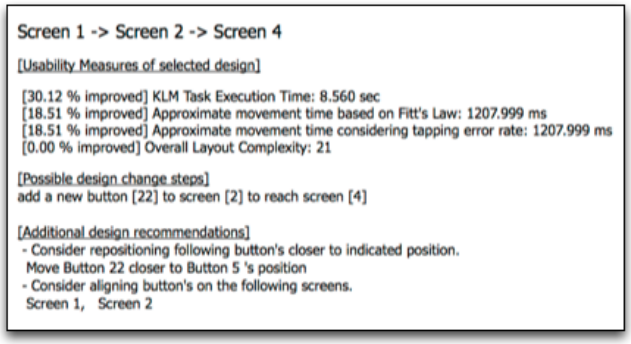


Figure 6.6 Zipcar: Reserve a Zipcar near the current location



(a) Design variation list



(b) Recommendations listed from selected variation

Figure 6.7 Facebook: Design variations and recommendations

6.1.2.3 Design Rationale

The second part of the experiment was designed to assess the effectiveness of my approach for analyzing design rationale. In a software design process, a design rationale is a set of reasons for design change decisions. The scope of design rationale includes keeping track of design assumptions, design constraints, and design decisions. It is considered particularly important for maintaining and verifying the system and software architecture (Tang, Jin, & Han, 2007). This experiment was conducted to investigate how the model transformation approach might contribute to design rationale.

In this part of the experiment, I presented three sets of "before" and "after" versions of actual application designs to the participants: the AngryBirds[9] application (Figure 6.10), the Bible.is[10] application (Figure 6.11), and the Scrabble[11] application (Figure 6.12). These designs were collected by keeping track of design version changes. Collecting designs of "before" and "after" designs is not trivial, as the distributors of an application typically do not retain an old design for continuing public use. However, keeping track of design versions per each change is important for design rationale. For instance, as shown in Figure 6.12, the Scrabble application design improved in the later version, assuming that the main goal of the application (i.e., the most frequently executed task) is to play a new game. The number of taps as well as the task execution time decreased from one version to the

[9] Angry Birds is a mobile game application developed and distributed by Rovio Entertainment Ltd. This is a game using the destructive power of the Angry Birds character by managing power and angle of a slingshot to destroy pig's castles. Screen shots used in Figure 6.10 (a) were captured from a mobile version (iPhone 4) of AngryBirds Seasons edition in 2012. Screen shots used in Figure 6.10 (b) were captured from a mobile version (iPhone 5) of AngryBirds Lite edition in 2014.

[10] Bible.is is an application used for reading and listening to different versions of the bible, developed and distributed by Fait Comes by Hearing. Screen shots used in Figure 6.11 (a) were captured from a mobile version (iPhone 4) in 2012. Screen shots used in Figure 6.11 (b) were captured from a mobile version (iPhone 5) in 2014.

[11] Scrabble is a mobile game application developed and distributed by Electronic Arts Inc. Scrabble is a word game that involves placing letter tiles on a board to form words. Screen shots used in Figure 6.12 (a) were captured from a mobile version (iPhone 4) in 2010. Screen shots used in Figure 6.12 (b) were captured from a mobile version (iPhone 5) in 2011.

next (the number of taps to play a new game decreased from four to three), indicating an improvement in usability in the 2011 version (Figure 6.12(b)). However, in the most recent version of this application (Figure 6.8), the number of taps as well as the task execution time has increased. The usability of the most recent design has arguably. If the design rationale of past versions of the design had been maintained, this apparent mistake (a "mistake" in the sense of decreasing efficiency) may not have been made.

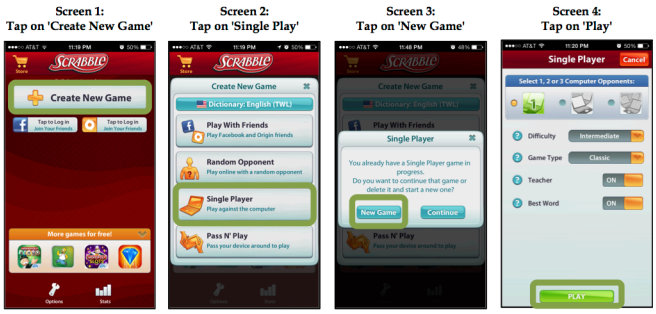
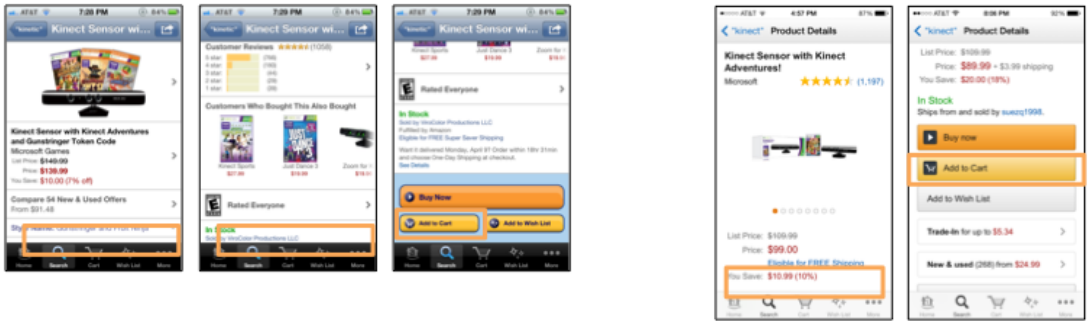


Figure 6.8 Scrabble design sequence in 2014

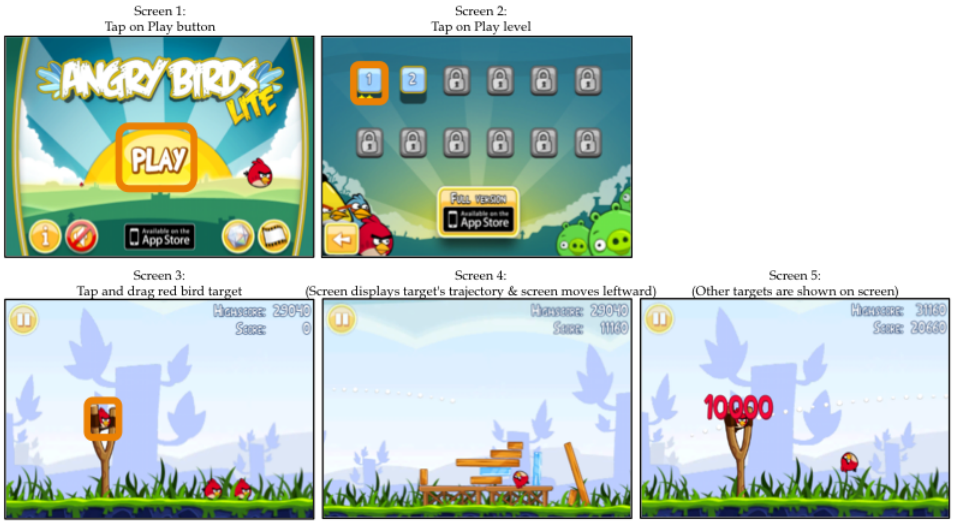


(a) Design in 2012

(b) Design in 2014

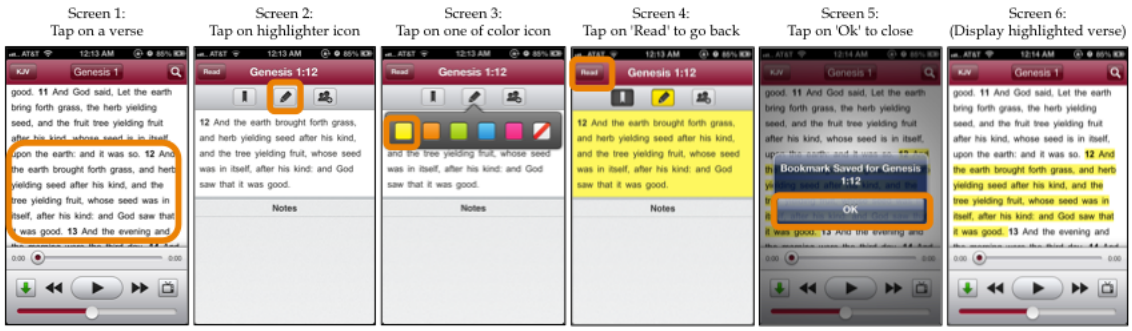
Figure 6.9 Amazon: Before and after versions

(a) Design in 2012

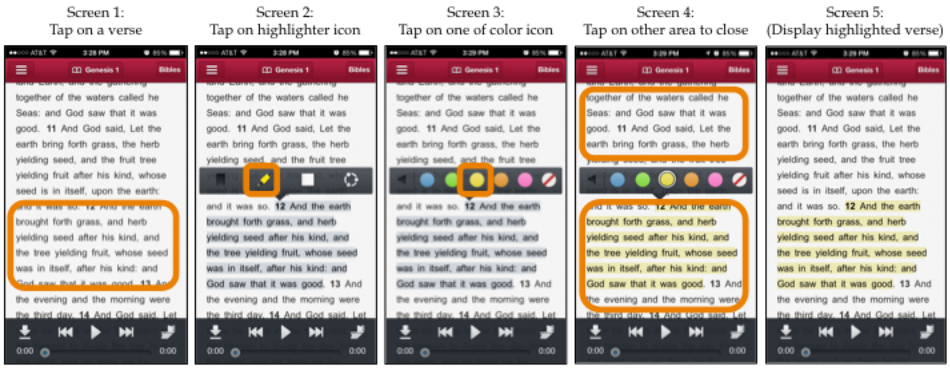


(b) Design in 2014

Figure 6.10 AngryBirds: Before and after versions

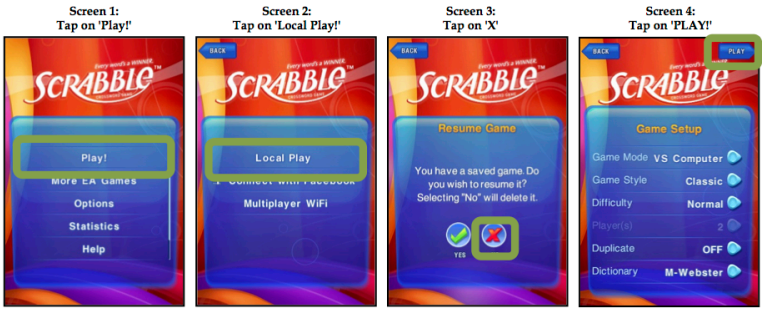


(a) Bible.is: Design in 2012



(b) Bible.is: Design in 2014

Figure 6.11 Bible.is: Before and after versions



(a) Design in 2010

(b) Design in 2011

Figure 6.12 Scrabble: Before and after versions

For each pair of designs, I asked participants to describe the differences between the two versions relevant to the efficiency of the designs, and to explain the design rationale behind the changes. The purpose of the rationale component of the experiment was to analyze how well model transformation

approach could automatically analyze the design rationale compared to the results produced by the participants. Below are the instructions for this part.

> *In the following problems, older and newer versions of mobile applications will be shown for comparison. Three designs will be presented. Analyze how the usability of the older design has been improved in the newer design; this is the design rationale.*
> *1. Summarize design changes and describe design rationales.*
> *(a) Summarize the design changes.*
> *(b) Specify the design rationale (reasons supporting the design decisions).*
> *2. Save and submit.*

To help participants understand the context, I provided an example before participants started on the actual problems. Example answers regarding design differences and design rationales with "before" and "after" design of Amazon[12] application design (Figure 6.9) are as follows.

[12] Amazon is an application used for online shopping, made and distributed by AMZN Mobile LLC. Screen shots used in Figure 6.9 (a) was captured from a mobile version (iPhone 4) in 2012. And screen shots used in Figure 6.9 (b) was captured from a mobile version (iPhone 5) in 2014

1. Reduced distance (number of swipes) in moving from kinect product to the 'add to cart' button.

In the older version, from kinect product screen you and swipe screen upward twice to the 'add to cart' button. In the new version, the 'add to cart' button is placed closer to the product description. The designer has reduced finger movement distance (number of swipe), so as to reduce movement time (or task execution time). Reducing total execution time improves the efficiency aspect of usability of the given task.

2. Larger 'add to cart' button.

The 'add to cart' button is larger in new design. Making icons/buttons larger should decrease the possibility of missing a tap on that button and will also reduce the movement of the finger to it from a previous position.

6.1.3 Procedure

A Web address was provided to students to participate in the experiment. All the materials were presented via an online survey tool, Qualtrics. A within-subjects design was adopted in order to compare results of the Traditional Condition with results of the Transformation Condition. A within-subjects design was chosen to eliminate confounds that might occur due to differences in ability between groups of participates. The Rationale component of the experiment was conducted before the Design Improvement component to reduce interference or learning effects.

Once participants clicked on the provided link to start the experiment, an informed consent form was provided and demographic information was collected. Students were allowed to opt out of the experiment as participants, but because the exercise had educational value they would have had to carry out the tasks as a regular assignment otherwise. Figure **6**.13 shows the overall ordering of materials for this experiment after informed consent was provided.

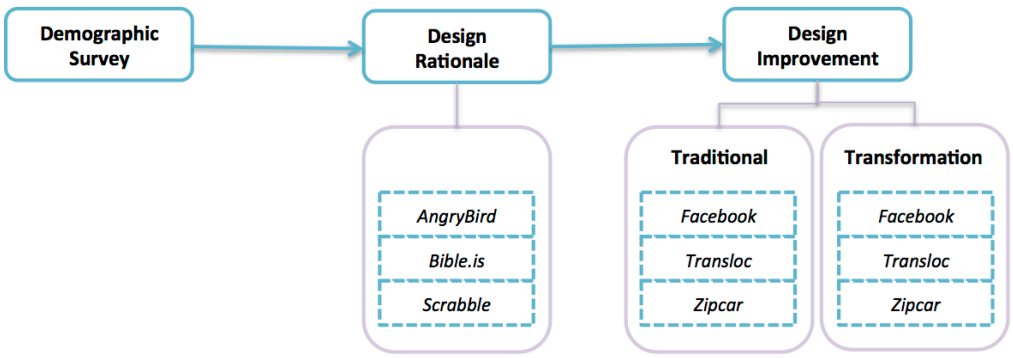


Figure 6.13 Experiment procedure

6.2 Results

Seventy-six students submitted responses for the experiment. I excluded data collected from students who completed only the demographic survey but not the main experiment. Qualtrics[13] was used to collect demographic data and to present the problems for participants to solve, but the results they produced were collected through a dedicated submission system. Because of this arrangement, some students were able to go through the experiment multiple times and submit only one set of outputs. Because of the risk of a learning effect from the prior participation, I excluded data collected from students who participated in the experiment multiple times. This resulted in 59 participants' data being used in the analysis. Note that 0.05 is used as a significance (or alpha) level for the statistical analyses.

[13] An online survey service provided at http://www.qualtrics.com

6.2.1 Demographics

Table 6.1 shows the basic demographic characteristics of participants. 90% of the participants were students majoring in computer science. This indicates that most of the participants had knowledge of developing and designing software. Based on their answers to additional survey questions, 80% of the participants had more than a year of experience in developing or designing user interfaces. Fifteen (25%) of the participants had more than four years of experience. I could expect that participants recruited had general knowledge about the process of developing and designing user interfaces.

Moreover, 59% of the participants had experience in developing a mobile application. 98% of the participants reported they use a handheld touch device such as a mobile phone regularly, indicating that the participants were familiar with using and understanding the interface designs and tasks presented in the experiment. One question asked participants to specify the names of usability models they were aware of. 98% of them were aware of more than one of usability evaluation model, from the set of Fitts' Law, the Keystroke Level Model (KLM), GOMS Model, ACT-R, Hick's Law, and the Layout Complexity model. Specifically, of the models used in the experiment (i.e., Fitts' Law, KLM, and the layout complexity), 93% reported they knew Fitts' Law (used in the *Traditional* Condition), 59 % reported they knew of two models, and 25% reported they knew all three models. Participants were not experienced modelers, but they had enough knowledge to understand the conditions and perform the required tasks by using information provided from the models.

Table 6.1 Demographic characteristics of student participants

	M	*SD*	*Range*
Age	25.86	4.93	20 -44
			Frequency
Gender	Male		61%
	Female		39%
Major	Computer Science		92%
	Engineering		5%
	Not Specified		3%

6.2.2 Design Improvement

This component of the experiment had three application designs for participants to improve, in two conditions. This required participants to submit six new (improved) designs (two for each application). However, not all of the participants submitted all designs. To analyze data collected from this session, I excluded incomplete submissions. The total number of participants considered for this analysis is 54. This is 162 designs for each condition, 108 designs for each application, and 324 designs in total. Results were analyzed to answer two core research questions:

- Do participants use the recommendations provided by the model transformation approach to improve the usability of the designs they are given?
- Does following a recommendation actually improve an interface design?

Evidence to answer these questions speaks to the effectiveness of my approach. Effectiveness, in this context, can be interpreted as "If the information produced by the model transformation approach is provided, as in the Transformation Condition, it will be effective in improving the usability of an interface during an iterative interface design and development process."

For analysis, I used KLM execution time as the main metric for measuring efficiency (a core aspect of usability) and categorized the type of information that participants used for improving usability. I did this for designs for both the Traditional and Transformation Conditions. I first determined whether the original functionality of the given design was maintained in both designs. Next I analyzed whether both designs fall into same category of design variation. In addition, I checked whether the design from the Transformation Condition consistently followed the recommendations provided by the system. Based on this data, I classified the use of the information in the Transformation Condition as follows. Recall that when the participant chooses a specific design variation, the system provides a recommendation for how to modify the original design such that it matches the design variation.

- **No use of the recommendation**: The participant did not follow the recommendation provided.
- **Incorrect use of the recommendation**: The participant followed the recommendation, but the functionality of the improved design did not match the original design.
- **Correct use of the recommendation**:
    1. The functionality of the design produced in the Traditional Condition is correct, and the participant adopted a recommendation based on one of the design variations.
    2. The KLM execution time changed from the design produced in the Traditional Condition, again through the adoption of a recommendation based on one of the design variations.
    3. The Fitts' Law movement time and/or the layout complexity metric improved by the recommendations based on Fitts' Law, Parhi et al.'s error model, or the layout complexity models.

In the remainder of this section I walk through the details of my analysis.

6.2.2.1 Do participants use the recommendation from the Transformation Condition?

This section illustrates the analyses of evidence for the first research question.

Based on a two-way analysis of variance with Condition and Application as the independent variables and KLM execution time as the dependent variable, the UI designs used for this experiment differ from each other enough to have effect on the solution and the Condition influences the usability of an improved design (Appendix A.1.1).

Table 6.2 Frequency of designs by use of the recommendation

Recommendation Usage Type	*Frequency*	*%*
No use	46	28.4
Correct use	113	69.75
Incorrect use	3	1.85
Total	162	100

As Condition was one of the main factors in the experiment, I analyzed whether the recommendation provided in the Transformation Condition was actually used when participants improved their designs.

Table 6.2 shows the frequency of the new designs from Transformation Condition, broken down based on use of the provided recommendation. Based on this data, approximately 70% of the designs used the information correctly. With this data, a Chi-Square test rejects the null hypothesis that the

categories of the Recommendation Usage Type occur with equal probabilities ($p < .001$). There is sufficient evidence to say that the proportions of designs in each category are different.

Figure 6.14 shows a chart of the number of new designs per each participant that used the recommendation in improving a design in the Transformation Condition. The x-axis represents the number of new designs created by the participants that applied the recommendation provided (0 indicates they did not use the recommendation in any of the designs; 3 indicates they followed the recommendation for all three of the designs). The y-axis represents the number of participants for each number. Five (9 %) of the participants didn't use the recommendation for any of their designs whereas 27 (50%) of the participants used the recommendation (incorrectly as well as correctly) for all three of their designs. 71% of the participants used the recommendation for at least one of their designs (Chi-Square Test: $p < .001$).

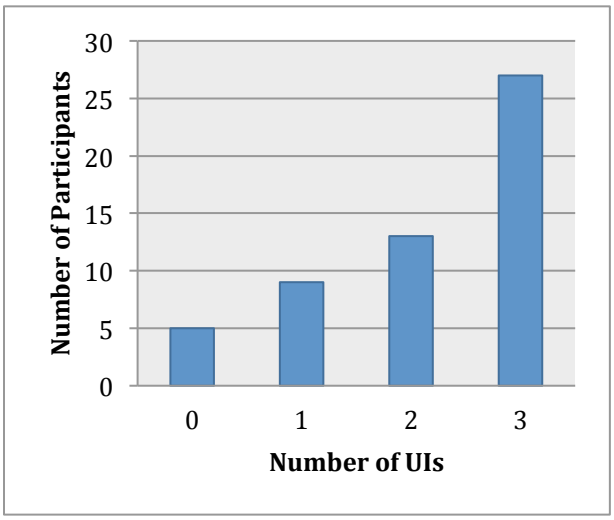


Figure 6.14 Number of improved designs per participant using the recommendation

Further analyses for the first research question are given in Appendix A.1. The ratio of using recommendations in the Transformation condition based on the presentation ordering of UIs was same for all orderings. The number of participants who used the recommendation in their earlier designs but not in their later ones was small. These results indicate that the presentation order of the design did not influence the use of the recommendation.

6.2.2.2 Is the recommendation from the Transformation Condition actually useful?

This section describes the analyses of evidence for the second research question. Participants were asked to improve a given design in two consecutive phases, first in the Traditional Condition and second in the Transformation Condition. My hypothesis was that even though the participants already had made decisions about possible design improvement based on their own knowledge and with information from the traditional model-based approach, they would be able to improve their design further with the recommendation from the model transformation approach.

Usability Difference between the Conditions

Figure 6.15, Figure 6.16 and Figure 6.17 show the distribution of the number of new designs (y-axis) falling into different KLM execution time values (x-axis) for each Condition by Application.

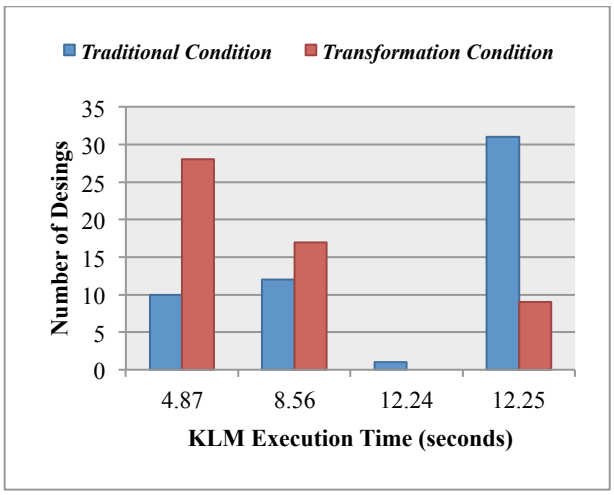


Figure 6.15 Facebook: Distribution of KLM execution times of improved designs

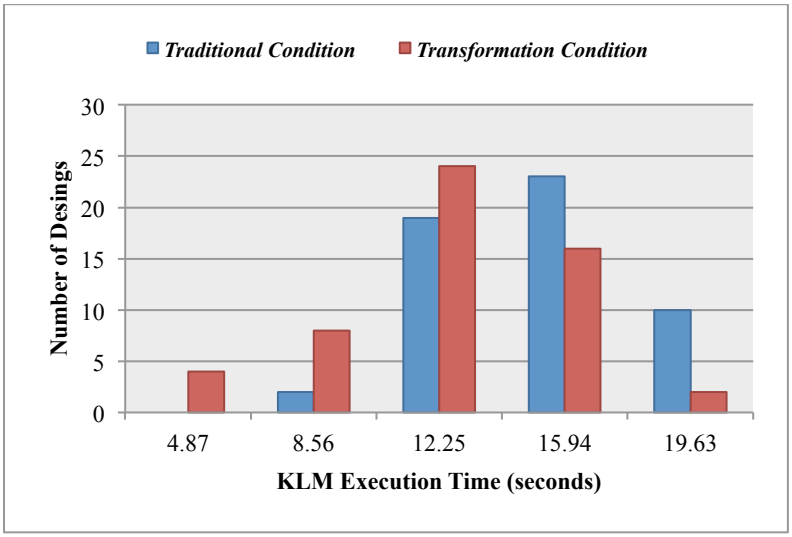


Figure 6.16 TransLoc: Distribution of KLM execution times of improved designs

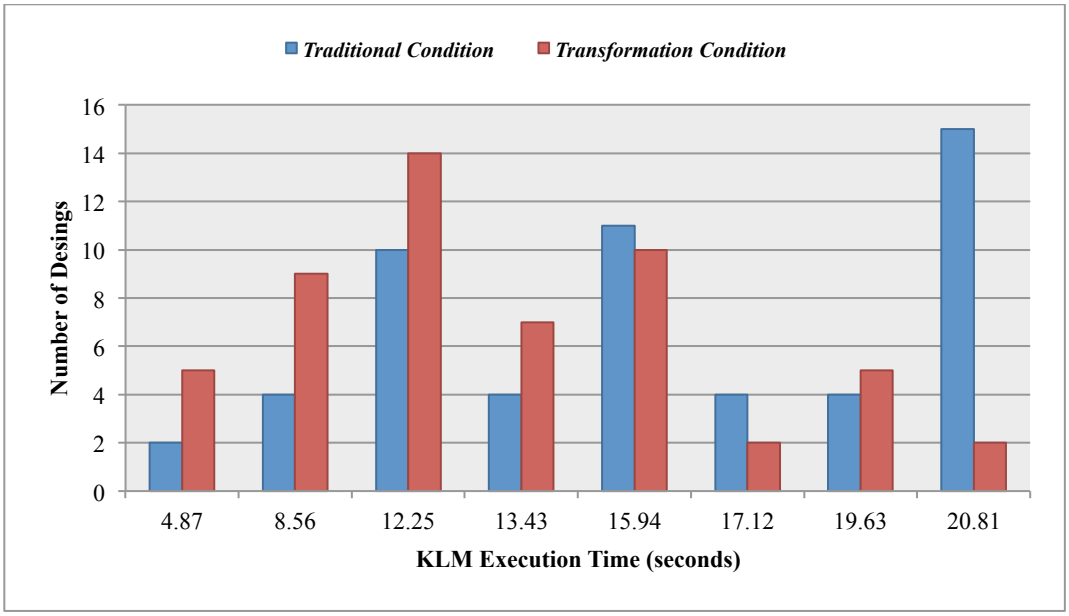


Figure 6.17 Zipcar: Distribution of KLM execution times of improved designs

Table 6.3 Summary statistics of KLM execution times

Application	***Traditional Condition***		***Transformation Condition***	
	Mean	*SD*	*Mean*	*SD*
Facebook	10.06	2.91	7.26	2.79
TransLoc	15.05	2.95	12.52	3.5
Zipcar	15.83	4.43	12.97	4.3

Table 6.3 shows summary statistics of KLM execution time of the improved designs for each Condition. To test the difference between the two means of KLM execution time, I performed a matched-paired *t*-test for each application design. The result shows that there is a significant difference between the means in the Traditional and Transformation Conditions for all three designs ($n = 54$), for each Application: Facebook: $t = -6.78$, $p < .0001$; TransLoc: $t = -6.32$, $p < .0001$; Zipcar:

$t = -5.85$, $p < .0001$). The mean of the Traditional Condition was larger than that of the Transformation Condition in each case. This indicates that the design of the Transformation Condition had better efficiency. Thus, the result provides evidence that interface designers or developers can improve the efficiency of a design by using the recommendations produced by the model transformation approach even after following the traditional approach.

Usability Difference Ratio between designs in the *Transformation* Conditions

In addition, I analyzed whether the effect of model information in the Transformation Condition was consistent among all designs. Because application designs were qualitatively different, comparing the times directly would not produce a fair comparison.

Table 6.4 KLM execution time difference ratio overall in the Transformation Condition

Application	*Num*	***Transformation Condition***	
		Mean	*SD*
Facebook	54	0.24	0.25
TransLoc	54	0.16	0.19
Zipcar	54	0.16	0.22

I computed the KLM execution time difference ratio between the designs of the Traditional and Transformation Conditions and analyzed the results using a one-way analysis of variance. (Table 6.4 shows the mean and standard deviation of the KLM execution time difference ratio between designs from both conditions.) The null hypothesis for this analysis was 'The means of the KLM execution time difference ratio are the same across all UIs'. The analysis of variance did not reject the null

hypothesis ($F = 2.2466$, $p = .1091$). This provides evidence that the effect of using the recommendation in the transformation approach is consistent across different UIs.

Further analyses for the second research question are given in Appendix A.2. The result of one analysis is that the recommendation was particularly helpful for designs that were not significantly improved in the Traditional Condition. Another analysis shows that the participants tended to select design variations with shorter execution times in the Transformation Condition compared to their selection in the Traditional Condition.

6.2.3 Design Rationale

Design rationale can be mapped to the model transformation step (that is, the model is transformed by transformation operators that should improve the usability of the user interface for which the model has been constructed). I interpret design rationale, in this context, as a set of decisions based on the metrics produced by different interaction models. Design differences map to the redesign step based on design implications of the design variations and the recommendation. After a decision to change a design is made in the model transformation step, a new design is produced by following the recommendation. In line with the direction of my research, designers should be able to reduce errors of producing designs that are not consistent with a given design rationale.

I included all 59 participants' data in my analysis, because all submitted at least one design difference and design rationale pair. There were 11.47 pairs on average for each participant (AngryBirds: 3.54 pairs, Bible.is: 4.08 pairs, Scrabble: 3.84 pairs). I analyzed the data in two ways: (1) a description of the design difference (design changes), and (2) a description of the design rationale for the design changes. The focus was on the following exploratory question: Does the model transformation approach contribute to design rationale?

My analysis in this section is descriptive rather than to test hypotheses. While significant research has been devoted to the automated inference of design rationale for software in general (Regli, Hu, Atwood, & Sun, 2000), I am unaware of any benchmarks for inferring design rationale directly from user interface designs.

Table 6.5 Number of differences based on difference category by UI

Difference Category	AngryBirds		Bible.is		Scrabble		Total	
Functional flow	0	(0 %)	1	(0.4 %)	9	(3.96 %)	10	(1.5%)
Number of actions	54	(25.8 %)	61	(25.3 %)	4	(20.3 %)	161	(23. 8 %)
Screen design/color	69	(33 %)	81	(33.6 %)	46	(17.6 %)	190	(28 %)
Target design/color	28	(13.4 %)	25	(10.4 %)	40	(35.2 %)	133	(19.6 %)
Target position	7	(3.4 %)	50	(20.75 %)	80	(17.6 %)	97	(14.3 %)
Target size	37	(17.7 %)	18	(7.5 %)	40	(3.5 %)	63	(9.3 %)
Incorrect	14	(6.7 %)	5	(2 %)	4	(1.8 %)	23	(3.4 %)

Table 6.5shows the number of differences described by the participants based on Difference Categories. An evaluator defined the Difference Categories by going through students' descriptions of the design differences and categorizing each. The data shows that a difference in screen design or color (28%) is the most frequently detected by the participants. Differences in the number of actions to perform a given task (23.8%) and differences in a target icon design or color (19.6%) follow.

I expected that use of the recommendations provided by the model transformation approach would produce fewer incorrect design changes. Twenty-three (3.4%) out of 677 of the differences were incorrect. Among incorrect ones, 69% were due to a participant's misinterpretation of the design. For example, one student described one difference in the AngryBirds design as 'Automatic display of target's expected trajectory in the new design in Screen 4.' However, the trajectory is actually the real trajectory after the target has been shot, rather than the expected trajectory. The rest were vaguely

articulated (3%) or referred to differences in screen size (4%), which participants had been instructed to ignore.

Table 6.6 Differences covered by model transformation system

Difference Category	Total	Covered by Model Transformation System
Functional Flow	8	8 (100 %)
Number of actions	151	143 (95 %)
Screen design/color	164	0 (0 %)
target design/color	127	0 (0%)
target position	93	76 (82 %)
target size	56	56 (100 %)

Table 6.7 Number of design rationales based on design rationale category by UI

Design Rationale Category	AngryBirds		Bible.is		Scrabble		Total	
Better feedback	37	(17.7 %)	45	(18.7 %)	55	(24.2 %)	137	(20.2%)
Better look and feel	21	(10 %)	17	(7 %)	13	(5.7 %)	51	(7.5 %)
Better recovery from mistake	0	(0 %)	4	(1.7 %)	3	(1.3 %)	7	(1 %)
Better use of screen estate	0	(0 %)	9	(3.7 %)	5	(2.2 %)	14	(2.1 %)
Better visibility	29	13.9 %)	27	(11.2 %)	33	(14.5 %)	89	(13.1 %)
Reduce ambiguity	18	(8.6 %)	4	(1.7 %)	23	(10.1 %)	45	(6.6 %)
Reduce clutter	1	(0.5 %)	4	(1.7 %)	5	(2.2 %)	10	(1.5 %)
Reduce error	32	(15.3 %)	9	(3.7 %)	15	(6.6 %)	56	(8.3 %)
Reduce time	39	(18 %)	90	(37.3 %)	61	(26.9 %)	190	(28 %)
Incorrect	32	(15.3 %)	32	(13.3 %)	14	(6.2 %)	78	(11.5 %)

I also explored the proportion of differences identified by the participants that can be analyzed by my model transformation system. (I excluded incorrect differences as well as the differences associated with an incorrect design rationale). Because the current scope of the models is on measuring efficiency, excluding other aspects of usability, my system was not able to analyze the differences in screen design or color, or differences in target icon design or color. However, the system was able to

reproduce all or part of the differences in categories (Table 6.6) that could be represented with the interaction models embedded in the system.

Table 6.7 shows the number of design rationales described by the participants for each design difference they found. The data shows that design rationales based on reducing user execution time (28%) and providing better feedback (20%) are most frequently described.

I expected that use of the model transformation approach would produce fewer incorrect design rationale observations. Seventy-eight (11.5%) of 677 design rationale submissions were incorrect. Within the set of incorrect design rationales, 29% were in accordance with design differences that were incorrectly identified (if a design difference is incorrectly characterized, the inferred design rationale will also be incorrect); 14% of the design rationales given were not associated with the design difference specified. Twenty one percent described the design itself as the design rationale, and the rest were vaguely articulated (30%).

I also explored the proportion of design rationales identified by the participants that could in principle be produced by my model transformation system. Because the current scope of the models is on efficiency, my system is limited to analyzing part of the 'reduce error' (78.6% of cases covered) and the 'reduce time' (90.5 % of cases covered) categories of design rationale.

6.3 Discussion

To evaluate the effectiveness of the model transformation approach, I conducted a two-stage experiment. In first stage, I asked participant to improve a set of user interfaces under two conditions, with or without the recommendations from my model transformation system. The second stage was an exploratory study analyzing design rationale for different versions of a set of user interfaces.

My analysis answered two research questions and one exploratory question: 'Do participants use the recommendations provided by the model transformation approach to improve the usability of the designs they are given?', 'Does following a recommendation actually improve an interface design?', and 'Does the model transformation approach contribute to design rationale? '

There is evidence that the answer of the first question is yes: I showed that Condition is a main effect on the difference in the usability of the improved design. Approximately 70% of the designs used the recommendations provided in the Transformation Condition even after a process of improving usability following a more traditional method, in the Traditional Condition. Moreover, 91% of the participants used recommendations for at least one of their three designs. I also showed that the order of the design presented did not influence the likelihood of adopting recommendations.

I can also answer yes to the second research question: There were significant differences in usability between the improved design from the Traditional Condition and the Transformation Condition for all three of the applications; the mean usability measure of improved design from the Transformation Condition was higher (i.e., smaller KLM execution time). The effect (i.e., usability difference ratio) of using the recommendation in the Transformation Condition was consistent across all designs. Additionally, the recommendation from the Transformation Condition was particularly useful when the interface was not improved significantly in the Traditional Condition. From the distributions of selected design variation category, I can further see that the designs using the recommendation in the Transformation Condition are in the category with smaller KLM execution times, compared with the category selected in the Traditional Condition.

In the second stage of the experiment, I explored the use of model transformation approach to infer design rationale. I first analyzed participants' descriptions of design differences, categorizing each for

two versions of the design. The model transformation approach was able to analyze 41.8 % of the differences that could be represented with the interaction models. Those categories include the difference in *functional flow*, *number of actions*, *target position*, and *target size*. I then analyzed the design rationale categories. The model transformation approach was able to analyze 78.6 % of cases in the *reduce time* category and 90.5% of cases in the *reduce error* category. I also expect the model transformation approach may be able to reduce problems of identifying incorrect design differences (3.4%) as well as incorrect design rationales (11.5%). From these results, I could also answer yes to the third question, by saying that there are possibilities to use the model transformation approach to contribute to design rationale.

Although I was able to collect evidence to validate the effectiveness of the model transformation approach through this experiment, limitations remain.

In the first stage, on Design Improvement component, I fixed the ordering of the conditions to be the traditional condition first and the transformation condition second, using the same set of designs. This may have produced a test re-test effect, in that the second exposure to the same design could itself have been the factor determining performance. In addition, there is the possibility of a diffusion effect: because the study was done online, the participants may had the chance to talk about their improvements in the first condition with each other, which might have had an impact on their performance in the transformation condition. These effects are threats to the internal validity of the study. I used a small number of UIs for the experiment (three in each of the stages of the experiment, six in total). Even though all six interfaces were different, more designs with varying complexity need to be tested experimentally to produce stronger evidence. I am using KLM execution time as the measurement for the usability of the improved design and to evaluate effectiveness. However, effectiveness can be defined using other metrics as well. For instance, comparing the average time the

participants spend in each condition for redesigning the interface, comparing the performance with subjective improvements made by design experts, or comparing the time spent on redesign with that of another model-based tool (e.g., CogTool) are all possibilities. More studies measuring multiple aspects of effectiveness are necessary to increase the generality of the work. In addition, a between-subjects experiment in which participants are assigned to conditions based on their modeling experience would increase external validity of my work.

7 CONCLUSION

In the process for developing a usable system or design, the evaluation of the interface is done either through user testing or model-based approaches. Model-based techniques are preferred in the earlier stages of the development to predict usability quickly and easily before user testing (Kieras, 2002). The traditional model-based approach, by constructing interaction models for given task and design, is able to provide evidence for making decisions about whether the UI needs to be redesigned to improve usability. However, the redesigning task is done by the designer based on the designer's knowledge; there is no guarantee that the new interface design has better usability. The designer has to construct interaction models to evaluate the usability of the new interface. Inspired by the fact that a constructed interaction model contains design information related to the usability, I proposed a model transformation approach for a model-based usability improvement process.

7.1 Summary

The goal of this research was to provide a framework to support the model transformation approach for improving the usability of a design during the iterative user interface development process. There is prior work on model-based evaluation that focuses on constructing appropriate models to evaluate usability ('model' in this context focuses on user performance) and model-based user interface generation ('model' in this context focuses on the user interface). However, there is no work on connecting these two areas by providing recommendations that directly guide the designer to redesign a user interface such that usability is improved.

In this dissertation, I first conducted a study that replicates John's study (John, 2011) on a model-based approach with CogTool, for evaluating the interface as well as for providing design recommendations. The findings from this study are motivation for my work (Chapter 3).

I then presented a model transformation approach that can produce design recommendations that directly lead to changes that improve usability. This approach is different from a traditional approach in automatically traversing possible design alternatives to improve a design. A model state is defined to present multiple interaction models that quantify different aspects of user behavior or design. The action graph formalism is used to incorporate these multiple interaction models. Three model transformation operators are defined in terms of action graphs (i.e., add an edge, remove an edge, remove a vertex). With these model states and operators, a general search algorithm is used to explore design variations. During the exploration, information for design recommendation is collected (Chapter 4).

The model transformation approach was implemented as a system with three components: a UI representation module, a model transformation module, and a design implications module. In the UI representation module, the Design Component Model is defined to incorporate the interaction models used in the system. In the model transformation module, a search algorithm is implemented to perform model transformation operations with model states and model transformation operators. This module also collects information produced in the exploration of the search space. Collected information is used for generating recommendations in the design implication module (Chapter 5).

Using the system I have implemented, I conducted empirical studies to demonstrate the effectiveness of the approach. A two-stage experiment was designed. The first stage examined how participants improve user interfaces under two conditions (with and without UI recommendations provided), and the second stage analyzed design rationale between different versions of UIs. The experiment produced evidence indicating that the participants tended to use the provided recommendations and that the usability of the user interface designs actually improved when they followed the

recommendations. In addition, we saw a possibility to contribute to design rationale through the model transformation approach (Chapter 6).

7.2 Future work

Although we have presented a new approach to improve the traditional approach to model-based usability evaluation for improving design, there are further issues to investigate. I first need to improve the effectiveness of this approach. An efficient scheme to manage the state space is required to handle designs that require larger number of taps to reach a goal. As mentioned in Section 5.4.1, the search space grows exponentially. Managing search space based on unique goal paths may increase efficiency. Also, more investigation of constraints on a given design, to support conflict checking and constraint prioritizing, will not only contribute to efficiency (excluding candidate states by the constraints specified) but may also produce improved recommendations that consider contextual information. Work on defining constraints should also link to refinements or extensions of the representation of the Design Component model, to embed contextual information of the design. Further work on improving the layout algorithm for automatically optimizing design components, considering the constraints of the design, should also produce more plausible guidance to designers. Although I have described a framework to incorporate multiple interaction models, more work is required for more sophisticated interaction models (e.g., models that can also analyze perception or cognition, such as GOMS (Card, Moran, & Newell, 1983), (Gray, John, & Atwood, 1993), (John & Kieras, 1996a), (John & Kieras, 1996b), (Kieras, 1999) and ACT-R (Anderson, Bothell, Douglass, Lebiere, & Qin, 2004)), (ACT-R Research Group, 2002). Supporting such cognitive models will enable more sophisticated levels of recommendation. To improve the efficiency of this approach, a *usable* tool that supports model transformation approach also needs to be developed. In this dissertation, I only show one possible system that supports model transformation; it does not consider

the usability of the system as a design tool. Developing such a tool that helps designers in practice is future work. My long term goal is a tool that is easy to use during the design improvement process and efficient enough to reduce the current time spent on improving a design.

BIBLIOGRAPHY

ACT-R Research Group. (2002). Retrieved 2014, from ACT-R: http://act-r.psy.cmu.edu

Anderson, J. R., Bothell, D., Douglass, M. D., Lebiere, C., & Qin, Y. (2004). An integrated theory of the mind. *Psychological Review , 11* (4), 1036-1060.

Bellamy, R., John, B., & Kogan, S. (2011). Deploying CogTool: integrating quantitative usability assessment into real-world software development. *Software Engineering (ICSE), 2011 33rd International Conference* (pp. 691-700). IEEE.

Bellamy, R., John, B., & Kogan, S. (2011). Deploying CogTool: Integrating Quantitative Usability Assessment into Real-world Software Development. *Proceedings of the 33rd International Conference on Software Engineering* (pp. 691-700). ACM.

Byrne, M. D. (1994). Automating interface evaluation . *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 232-237). ACM.

Card, S. K., Moran, T. P., & Newell, A. (1980). The Keystroke-level Model for User Performance Time with Interactive Systems. *Communications of the ACM , 23* (7), 396-410.

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

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

Dix, A., Finlay, J. E., Abowd, G. D., & Beale, R. (2003). *Human-Computer Interaction.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement . *Journal of Experimental Psychology , 47* (6), 381-391.

Gajos, K. a. (2004). SUPPLE: Automatically Generating User Interfaces. *Proceedings of the 9th International Conference on Intelligent User Interfaces* (pp. 93-100). ACM.

Galitz, W. (2002). *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques.* John Wiley & Sons, Inc.

Grammenos, D. A. (2000). Integrated support for working with guidelines: the Sherlock guideline management system. Interacting with Computers. *Interacting with Computers , 12* (3), 281-311.

Gray, W. D., John, B. E., & Atwood, M. E. (1993). Project Ernestine: A validation of GOMS for prediction and explanation of real-world task performance. *Human-Computer Interaction , 8* (3), 237-309.

Heim, S. (2007). *The resonant interface: HCI foundations for interaction design.* Boston: Addison-Wesley Longman Publishing Co., Inc.

Hertzum, M., & Jacobsen, N. E. (2001). The evaluator effect: a chilling fact about usability evaluation methods. *Int. Journal of Human-Computer Interaction* , 421-443.

Holleis, P., Otto, F., Hussmann, H., & Schmidt, A. (2007). Keystroke-level model for advanced mobile phone interaction. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1505-1514). ACM.

Hong, K. W., & St. Amant, R. (2014). Novice Use of a Predictive Human Performance Modeling Tool to Produce UI Recommendations. *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems* (pp. 2251-2254). ACM.

Hong, K. W., & St.Amant, R. (2013). Integrating Action Graphs and Keystroke Level Model to Represent the Usability Improvement Process. *Graphics Interface.* ACM.

Ivory, M., & Hearst, M. (2001). The State of the Art in Automating Usability Evaluation of User Interfaces. *ACM Computing Surveys , 33* (4), 470--516.

John, B. (2011). Using Predictive Human Performance Models to Inspire and Support UI Design Recommendations. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 983-986). ACM.

John, B., & Kieras, D. (1996b). The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Transactions in Computer-Human Interaction , 3* (4), 320-351.

John, B., & Kieras, D. (1996a). The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI) , 3* (4), 320-351.

John, B., & Salvucci, D. (2005). Multi-Purpose Prototypes for Assessing User Interfaces in Pervasive Computing Systems. *Pervasive Computing , 4* (4), 27-34.

John, B., Prevas, K., Salvucci, D., & Koedinger, K. (2004). Predictive Human Performance Modeling Made Easy. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 455--462). ACM.

Kieras, D. (1999). *A guide to GOMS model usability evaluation using GOMSL and GLEAN3.* University of Michigan, Electrical and Computer Engineering Department, Ann Arbor, MI.

Kieras, D. (2002). Model-based evaluation. In J. Jacko, & A. Sears (Eds.), *The Human-Computer Interaction Handbook* (pp. 1139-1151). Mahwah, New Jersey: Lawrence Erlbaum Associates.

Kieras, D. (1988). Towards a practical GOMS model methodology for user interface design. (M. Helander, Ed.) *The Handbook of Human-Computer Interaction* , 135-158.

Kieras, D. (1993). *Using the Keystroke-Level Model to Estimate Execution Times.* The University of Michigan.

MacKenzie, I. S., & Buxton, W. (1992). Extending Fitts' Law to Two-dimensional Tasks. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 219--226). ACM.

Oracle. (2014). *FlowLayout (Java Platform SE8)*. Retrieved from http://docs.oracle.com/javase/8/docs/api/java/awt/FlowLayout.html

Parhi, P., Karlson, A. K., & Bederson, B. B. (2006). Target Size Study for One-handed Thumb Use on Small Touchscreen Devices. *Proceedings of the 8th Conference on Human-computer Interaction with Mobile Devices and Services* (pp. 203-210). ACM.

Puerta A. R., E. H. (1994). Model-Based Automated Generation of User Interface. *AAAI Proceedings* (pp. 471-477). AAAI.

Regli, W. C., Hu, X., Atwood, M., & Sun, W. (2000). A survey of design rationale systems: approaches, representation, capture and retrieval. *Engineering with Computers , 16* (3-4), 209-235.

Ritter, F., Baxter, G., Jones, G., & Young, R. (2001). User interface evaluation: How cognitive models can help. In J. Carroll (Ed.), *Human-computer interaction in the new millenium* (pp. 125-147). Reading, MA: Addison-Wesley.

Sears, A. (1995). AIDE: A step toward metric-based interface development tools. *In Proceedings of the ACM Symposium on User Interface Software and Technology* (pp. 101-110). ACM.

Sears, A. (1993). Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout. *IEEE Transactions on Software Engineering , 19* (7), 707-719.

Szekely, P. (1996). Retrospective and Challenges for Model-Based Interface Development. *Design, Specification and Verification of Interactive Systems '96* (pp. 1-27). Springer-Verlag.

Szekely, P., Luo, P., & Neches, R. (1993). Beyond Interface Builders: Model-Based Interface Tools. *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems* (pp. 383--390). ACM.

Tang, A., Jin, Y., & Han, J. (2007). A Rationale-based Architecture Model for Design Traceability and Reasoning. *Journal of System Software* , 918-934.

Thimbleby, H. (2013). Action Graphs and User Performance Analysis. *nternational Journal of Human-Computer Studies , 71* (3), 276-302.

Vanderdonckt, J. (1995). Accessing Guidelines Information with SIERRA. *Proceedings of Fifth IFIP TC 13 International Conference on Human-Computer Interaction INTERACT'95* (pp. 311-316). Chapman & Hall.

Wilson, M., Mackay, W., Chi, E., Bernstein, M., Russell, D., & Thimbleby, H. (2011). RepliCHI-CHI should be replicating and validating results more: Discuss. *Proceedings of the annual conference extended abstracts on Human factors in computing systems* (pp. 463-466). ACM.

Wobbrock, J. O., Cutrell, E., Harada, S., & MacKenzie, I. S. (2008). An Error Model for Pointing Based on Fitts' Law. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1613-1622). New York: ACM.

APPENDICES

Appendix A Additional Data Analysis of Design Improvement Study

A.1 Do Participants use the Recommendation from the Transformation Condition?

A.1.1 Factors causing a Usability Difference between the Conditions

This analysis is done to investigate whether the difference between the Traditional Condition and the Transformation Condition significantly affected the usability of the designs produced by participants. As a secondary factor, aside from Condition, I also considered Application, i.e., the specific design problem posed to the participants.

I performed a two-way analysis of variance with Condition and Application as the independent variables and KLM execution time as the dependent variable. The null hypotheses for this analysis were (1) there are no significant differences between the conditions with respect to KLM execution time, and (2) there are no significant differences between the applications with respect to KLM execution time. Analysis of variance rejects both hypotheses, indicating that the application (F = 85.3725, $p < .0001$) and the condition (F = 48.0766, $p < .0001$) are significant main effects. This means that the UI designs used for this experiment differ from each other enough to have effect on the solution. More importantly, the Condition (the type of information provided to the participant) also influences the usability of an improved design.

A.1.2 Use of the Recommendation by Presentation Order of the Designs

Since the application designs were presented in random order, I analyzed whether the order of presentation influenced the ratio of using the recommendation in the Transformation Condition. I counted the frequency of the Recommendation Usage Type by the order of designs. Fisher's Exact

Test (p = 0.83) shows there are no significant differences in the rate of each Recommendation Usage Type across all orderings. The ratio of using the recommendation was same for all orderings.

Further, I investigated whether there were participants who used the recommendation in earlier designs but didn't for later ones. Significant number of participants in this case may indicate that their experience with using the recommendation in the earlier designs was not satisfactory, such that they decided not to use the recommendation in the later designs. I counted the frequency of usage patterns. The result showed that one participant (1%) used information in the first UI but not for the later UIs. Three participants (5.6%) used information for the first and second UIs but not for the third one. The number of these cases is not large.

A.2 Is the Recommendation from the Transformation Condition actually useful?

A.2.1 Usability Difference by the Use of the Recommendations

I further investigated whether the participants' decisions to use the recommendation in the Transformation Condition (i.e., Recommendation Usage Type) depends on whether they were able to improve the design adequately in the Traditional Condition (resulting in no need to use information in the Transformation Condition).

I first tested for a difference in the KLM execution time difference ratio between the original design and the improved design in the Traditional Condition. A one-way analysis of variance was used with difference ratio as the dependent variable and whether the design used the recommendations in Transformation Condition as the factor. The null hypothesis was that 'There is no significant difference between the means when the participants used the recommendations in the Transformation Condition and when they did not'. Summary statistics of the data and a box plot are shown Figure A.1. The analysis shows a significant difference between the means ($F = 16.069$, $p < .00001$), rejecting the

null hypothesis. This can be interpreted as the participants who made a significantly greater ratio of improvement in the Traditional Condition tend to not use the recommendation in the Transformation Condition.

Use of Recommendations	*N*	*Traditional Condition*	
		Mean	*SD*
No use (0)	46	0.32	0.2
Use (1)	116	0.18	0.19

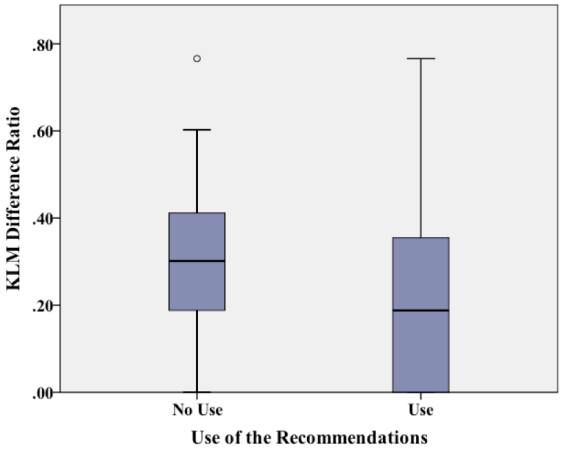


Figure A.1 Traditional Condition: KLM execution time difference ratio by use of recommendation

I did the same analysis with the difference ratio in the Transformation Condition, using the difference between the new design from the Traditional Condition and the new design from the Transformation Condition. A one-way analysis of variance was used with the difference ratio as the dependent variable and whether the design used the recommendations as the factor. The null hypothesis was that 'There is no significant difference between the means when the participants used the recommendations in the Transformation Condition and when they did not'. Summary statistics of the data and a box plot is shown in Figure A.2. The analysis shows a significant difference in the means ($F = 38.04$, $p < 0.0000001$), rejecting the null hypothesis. This indicates that the designs that used the

recommendations in the Transformation Condition produced a significantly greater improvement than those that didn't use the recommendation.

Use of Recommendations	*N*	*Transformation Condition*	
		Mean	*SD*
No use (0)	46	0.03	0.08
Use (1)	116	0.25	0.23

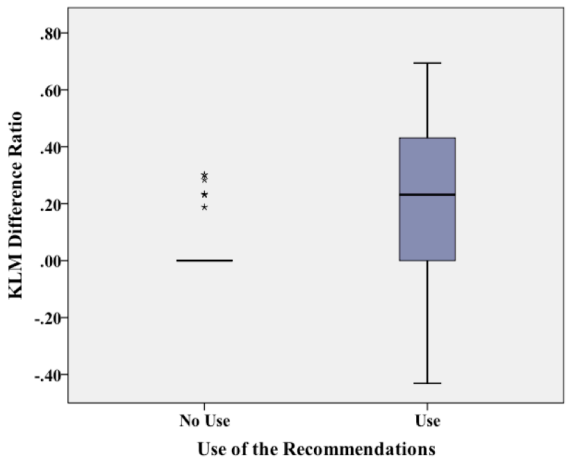


Figure A.2 Transformation Condition: KLM execution time difference ratio by use of recommendation

Lastly, I considered the overall KLM difference ratio, the difference between the original design and the new design from the Transformation Condition. I applied the same analysis technique. The null hypothesis was that 'There is no significant difference between the means of the difference ratio for both cases (used the recommendations in the Transformation Condition or not)'. Summary statistics of the data and a box plot are shown in Figure A.3. The analysis shows no significant difference in the means ($F = 2.29$, $p = 0.13$), which means that the null hypothesis cannot be rejected. This suggests that there is no significant difference in the improvement between the designs that did use the recommendations and those that didn't use the recommendations, considering both conditions as the whole process.

Use of Recommendations	*N*	*Both Conditions*	
		Mean	*SD*
No use (0)	46	0.34	0.19
Use (1)	116	0.4	0.21

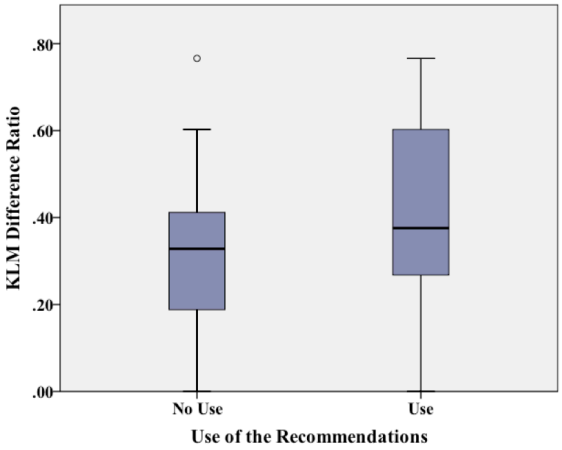


Figure A.3 Both conditions: KLM execution time difference ratio by use of recommendation

Based on these results I find evidence that the recommendations from the model transformation approach can be useful even after a designer has followed a traditional model-based approach, particularly when interfaces were not improved significantly in the traditional model-based approach.

A.2.2 Distribution of KLM Execution Time of Selected Design Variation

Information produced from the model transformation approach can generate possible design improvement plans (design variations) automatically, at least within the scope of models incorporated into the system. We can explore the distribution of KLM execution time of selected design variation category, with analysis based on the improved designs in both conditions, by whether the recommendation was used in the Transformation Condition. The x-axis of the distribution maps to the KLM execution time of selected design variation category provided in the Transformation Condition and the y-axis indicates the frequency of designs. As the design variation list is ordered in ascending order based on the number of screens of the goal path, larger design variation category number have

equal or larger KLM execution time. Note that as the list is composed of a set of permutations of screen sequences to reach the goal screen, a variation with same number of screen will have same KLM execution time. Table A.1 is the data table showing the number of improved Facebook designs that belong to each design variation category for each condition. This table particularly shows the cases of designs that didn't used the recommendations provided in the Transformation Condition.

Table A.1 Facebook: Design variation categories and design frequencies when recommendation not used

			***Traditional* Condition**	***Transformation* Condition**
Design Variation #	*Screen Sequence*	*KLM execution time (sec)*	*Number of designs*	*Number of designs*
1	Screen 1 -> Screen 4	4.87	6	6
2	Screen 1 -> Screen 2 -> Screen 4	8.56	3	3
3	Screen 1 -> Screen 3 -> Screen 4	8.56	2	3
4	Screen 1 -> Screen 2 -> Screen 3 -> Screen 4	12.25	4	3

To analyze the distribution difference of the KLM execution time for selected design variation categories, I created data tables such as Table A.1 for all application design in both conditions. I also created graphs for the frequency of the design by the KLM execution time of selected design variation.

The graphs in Figure A.4, Figure A.5, and Figure A.6 show the distributions of the KLM execution time of selected design variation categories of the improved designs from both conditions, specifically that didn't use the recommendations provided in the Transformation Condition.

As can be seen in the graphs, the distributions of the number of improved designs in the *Traditional* Condition and improved designs in the Transformation Condition do not show large differences. The selection of the design variation category in the Transformation Condition did not change a lot even when the recommendations were provided.

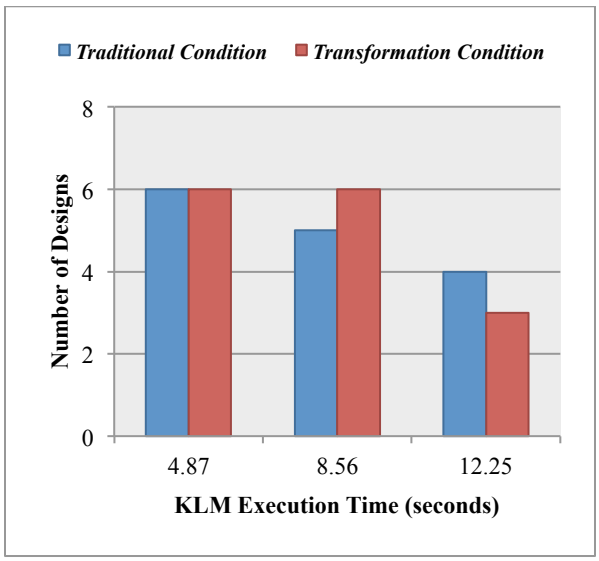


Figure A.4 Facebook: Distribution of KLM execution times when recommendation not used

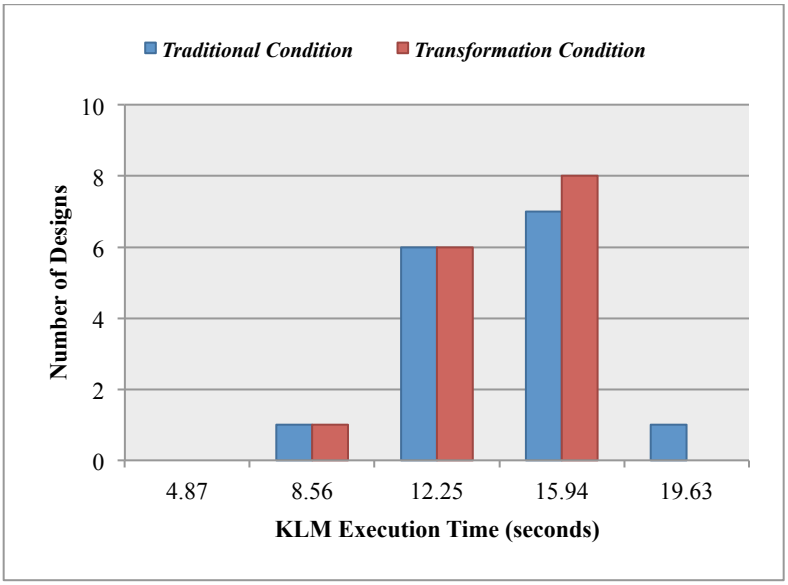


Figure A.5 TransLoc: Distribution of KLM execution times when recommendation not used

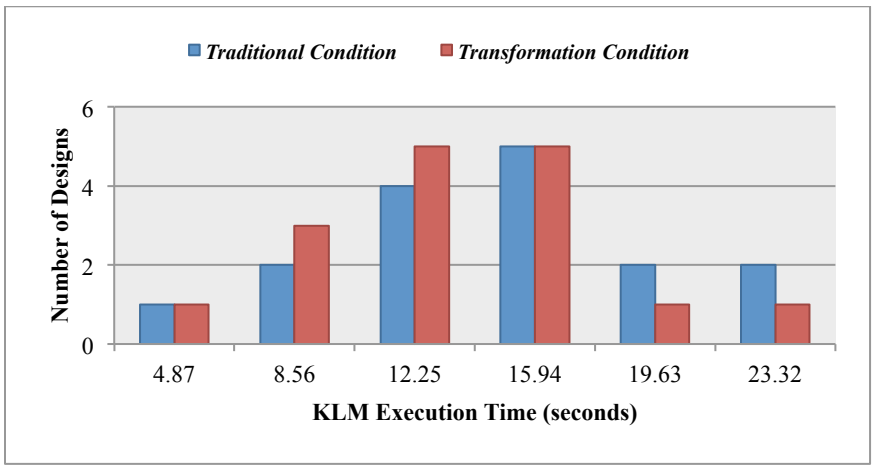


Figure A.6 Zipcar: Distribution of KLM execution times when recommendation not used

The graphs in Figure A.7, Figure A.8, and Figure A.9 show the distributions of the KLM execution times of selected design variation categories of the improved designs from both conditions, specifically those that used the recommendations provided in the Transformation Condition.

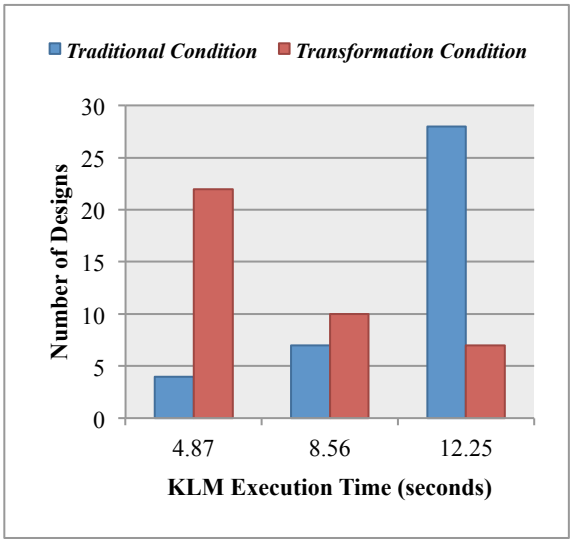


Figure A.7 Facebook: Distribution of KLM execution times when recommendation used

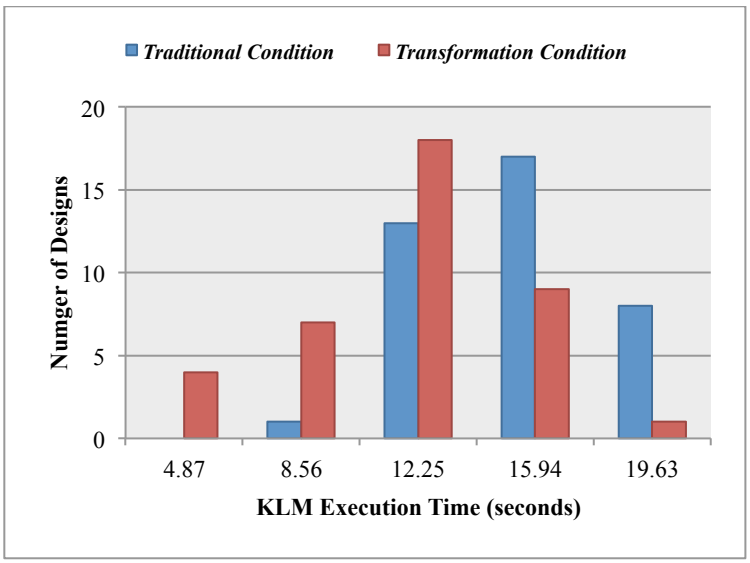


Figure A.8 TransLoc: Distribution of KLM execution times when recommendation used

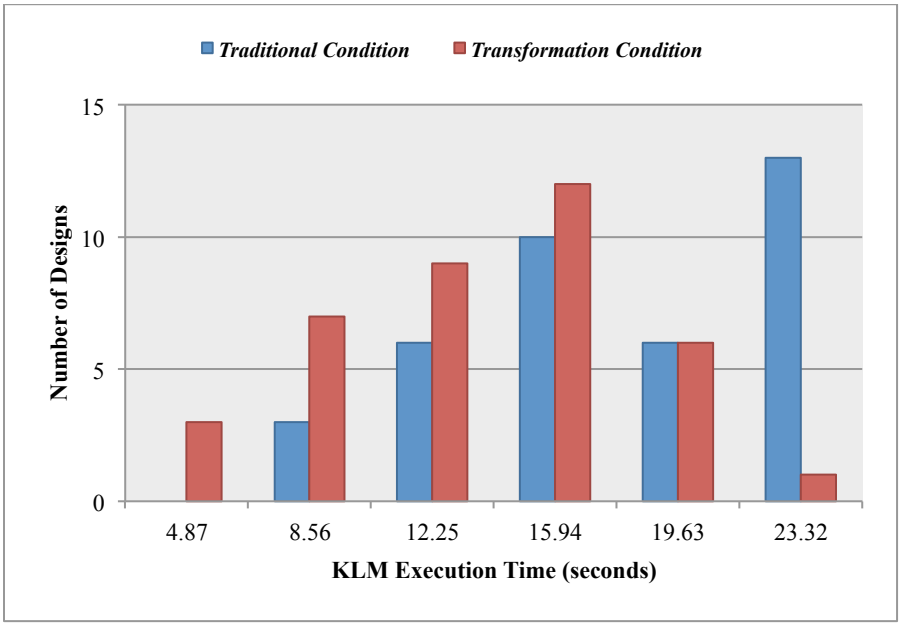


Figure A.9 Zipcar: Distribution of KLM execution times when recommendation used

The distributions in the graphs in Figure A.7, Figure A.8, and Figure A.9 do seem to show some differences in KLM execution time for the selected design variation. For the Facebook application (Figure A.7), the most frequently selected KLM execution time in the Traditional Condition is 12.25 seconds (the longest). However, in the Transformation condition, the KLM execution time of the most frequently selected design variations category is 4.87 seconds (the shortest). For the TransLoc application (Figure A.8), the frequency of improved designs belonging to shorter KLM execution time (from 4.87 seconds through 12.25 seconds) increased in the Transformation condition. The frequency of larger KLM execution times (15.94 seconds and 19.63 seconds) also decreased in the *Transformation* condition. For the Zipcar application (Figure A.9), the frequency of KLM execution times of the selected design variation category, with 23.32 seconds (the longest), decreased in the Transformation Condition. And the frequency of shorter KLM execution times (from 4.87 seconds to 15.63 seconds) increased in the Transformation Condition. Likewise, the KLM execution time of the selected design variation tends to decrease (the frequency of design variations with smaller KLM execution increased). This is more evidence for the model transformation approach being useful even after a designer has followed a traditional model-based approach.