Early Language Learning is a Good Model for Studying Early User Interface Learning

by

Erin Cristina Lester

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2005

Abstract

To date, the self-revealing interface has been the elusive holy grail of the user interface community. This research advocates the use of early language learning as a model for early user interface learning. This model can be used to reason about how users learn through exploration, and gain ideas as to how to design the implicit, online help needed to make a user interface self-revealing. The idea for this model came from a strong analogy between user interfaces and language. This analogy is based on fundamental similarities, and strengthened both by observations in a case study, and the general user interface literature. A case study of early exploratory user interface learning was done in the hopes of finding similarities between the learning of languages and interfaces. Although the study did reveal many similarities, which support the model, what was most interesting was their differences. Most notably, motherese, an important form of supportive feedback that is universally present in language learning, was missing in the user interface learning. Motherese is a distinct speech variant that is used by experienced language users in conversing with children. It helps to guide children towards an understanding of correct behaviours through acknowledgment, repetition, and correction of their utterances. An experiment was devised to evaluate an analogous type of instruction in the bootstrap learning of a novel user interface technique. The experiment validated the instruction's ability to shorten the initial learning period and ingrain new techniques better than un-aided exploratory learning. Motherese-style instruction meets the requirements for instruction that is self-revealing, and is firmly grounded by the strong analogy between language and user interfaces. The application of it to user interface learning is online and integrated within the actual context of the application. It is also demonstrative and non-verbal, giving users implicit instruction, and therefore does not suffer from the terminology or contextual switching issues that written instruction does.

Although a number of questions remain to be answered about the general applicability of motherese-inspired user interface instruction, the model presented has yielded the first empirically-based idea for designing self-revealing instruction. It is anticipated that future research using this model will help researchers to reason about both self-revealing instruction and new user behaviour.

Acknowledgements

This thesis has been made possible through the help, support and advice of numerous people.

First I would like to thank my supervisor Bill Cowan for sharing his great knowledge, ideas, advice and patience. He has taught me not only about user interfaces, but about life in general.

I thank my many friends and family members for their support, their love, and for believing in me. In particular, I thank Celine for all her advice and for always being there for me, especially when times were rough, Rob K. for letting me bounce ideas off him and drag him on our many runs, Tom, Dave and Rob W. for the great talks and distractions, and Elodie for her help and great ideas.

I would also like to thank all the members of the CGL for their support, ice cream (thanks Steve!) and being my family on campus, as well as the S05 CS789 class for their voluntary and enthusiastic participation in my study.

Lastly I thank my parents, mono, tigger, and Vlad for their undying love, support and needed distractions.

Trademarks

Contents

1 Introduction **1**

2 Background **6**

2.1 Motivation 6

2.1.1 Directions for the design of help resources 7

2.1.2 Problems with current interface learning supports 8

2.2 Self-Revealing Interfaces 11

2.2.1 Non-trivial self-revealing interfaces 12

2.2.2 Self-revealing interface elements 12

2.2.3 Are existing interfaces self-revealing? 14

3 The Language-Interface Analogy **15**

3.1 Previous References 16

3.2 Important Traits of Language 18

3.2.1 A communication and learning medium 18

3.2.2 Functional medium 19

3.2.3 Context dependent 19

3.2.4 Discrete combinatorial system 19

3.3 Interface Analogies 20

3.3.1 A communication and learning medium 20

3.3.2 Functional medium 21

3.3.3 Context dependent 21
3.3.4 Discrete combinatorial system 22
3.3.5 Summary 23
3.4 Early Language Learning 23
3.4.1 Production iteratively refines comprehension 23
3.4.2 Comprehension and production refinement is selective 24
3.4.3 Comprehension precedes production 25
3.4.4 Vocabulary size, utterance length and utterance diversity increase concurrently 25
3.4.5 Overgeneralization occurs transiently 25
3.4.6 Comprehension and production refinement occur without instruction 26
3.5 Motherese 26
3.5.1 Informative feedback 27
3.5.2 Conversation flow and control 27
3.5.3 Simplified forms 28
3.5.4 Sensitivity to learners' abilities 28
3.5.5 Phonetic differences 28
3.5.6 Summary 29

4 Case Study of Early Interface Learning 30
4.1 Goal 30
4.2 Method 31
4.2.1 Study setup 31
4.2.2 The user 31
4.2.3 The learning task 32
4.3 Observations 32
4.3.1 Production iteratively refines comprehension 32

4.3.2 Comprehension and production refinement is selective 34
4.3.3 Comprehension precedes production 34
4.3.4 Vocabulary size, utterance length and utterance diversity increase concurrently 35
4.3.5 Overgeneralization occurs transiently 36
4.3.6 Comprehension and production refinement occur without instruction 37
4.4 Conclusions 38
4.5 Lessons about Interface Learning 39
4.5.1 Production iteratively refines comprehension 40
4.5.2 Comprehension and production refinement is selective 40
4.5.3 Comprehension precedes production 41
4.5.4 Vocabulary size, utterance length and utterance diversity increase concurrently 41
4.5.5 Overgeneralization occurs transiently 41
4.5.6 Comprehension and production refinement occur without instruction 42
4.5.7 Summary 42

5 Motherese Experiment **43**
5.1 Overview 43
5.2 Motherese 44
5.2.1 Exploratory learning 45
5.3 Experiment 45
5.3.1 Goal 46
5.3.2 Task to be learned 46
5.3.3 The experimental procedure 48
5.3.4 Feedback 48

5.3.5 Apparatus 54
5.3.6 The subjects 54
5.3.7 Instruction 55
5.4 Results 55
5.4.1 Measurements 55
5.4.2 Hypotheses and statistical tests 59
5.4.3 Analyses 61
5.5 Conclusions 75

6 Discussion and Conclusions 77
6.1 Motherese 78
6.1.1 Implications of the experiment 78
6.1.2 Limitations of the experiment 78
6.1.3 Future studies of motherese in the interface 79
6.1.4 Interface motherese in practice 81
6.1.5 Other applications of motherese 82
6.2 Using the Early Language Learning Model 83
6.2.1 Implications of the model 84
6.2.2 Future directions 84
6.3 Conclusions 86

A Analogy Glossary 88

B Experiment Information Materials 90
B.1 Information Letter 90
B.2 Consent Form 91
B.3 Debriefing Letter 92

C Experiment Instructions 94

Bibliography **96**

List of Tables

5.1 Summary of learning period lengths for the two instruction types in the two blocks . . . 67

5.2 Regression analysis of the *TransferTime* in all mischord-free trials with successful transfers . . . 67

5.3 One-way ANOVA of *Instruction Type* on *Mischords* in the post-learning period . . . 69

5.4 One-way ANOVA of *Instruction Type* on *ClickTime* in the post-learning period . . . 70

5.5 Regression analysis of *Mischords* for the motherese instruction group compared to the average . . . 71

5.6 Regression analysis of *Mischords* for the control instruction group compared to the average . . . 71

5.7 Regression analysis of the post-learning period *TransferTime* for the motherese instruction group compared to the average . . . 73

5.8 Regression analysis of the post-learning period *TransferTime* for the control instruction group compared to the average . . . 75

A.1 Summary of the analogy between spoken language and the user interface . . . 88

A.2 Summary of the analogy between children's language learning and early un-aided exploratory user interface learning . . . 89

List of Figures

2.1 An example of online user-driven help: the Microsoft Agent 9

2.2 A pie menu 13

3.1 Context dependent mouse buttons in XFig 22

5.1 A button with motherese-style instruction 45

5.2 The chords used for drop and acquire actions and their effects . . . 47

5.3 The experiment interface 50

5.4 Correctional motherese-style instruction for the *acquire* action . . . 51

5.5 Correctional motherese-style instruction for the *drop* action 52

5.6 Reinforcement motherese-style instruction 53

5.7 Phases in a *successful* transfer 56

5.8 Phases in a *complete* trial 57

5.9 Average number of transfers completed per trial by each instruction type in block one 63

5.10 Average number of transfers completed per trial by each instruction type in block two 65

5.11 Average *TransferTime* 68

5.12 Average *Mischords* by trial for each instruction type in the post-learning trials 72

5.13 Comparison of the average *TransferTime* by trial for the two instruction types in the post-learning period 74

Chapter 1

Introduction

Automated telephone systems, which, like VCRs [1, p. 3], are famously unintuitive and difficult to learn [2, p. 17–23], and bicycles, which have a simple and intuitive design, have an important feature in common: in each interface users continue to learn, long after they have mastered the basic skills. This learning occurs naturally, while using the interface.

Most anyone can learn how to ride a bicycle. Once learned, riding skills are rarely forgotten. In addition, bicycle riding ability improves continually and is refined with practice. Most individuals learn to ride as a child, with a parent's hand at her back to catch her if she falters. But, although a parent provides support and says things like "pedal, pedal!" and gives reassurance like "don't worry I'm still here", he or she does not give explicit instruction. For example, he or she does not try to explain how to balance a bicycle for the simple reason that he or she cannot articulate the tacit knowledge he or she has. The child must learn by trial and error how to balance the bicycle vertically and propel it forward. Certainly, before ever sitting on a bicycle seat, a child has ideas about how a bicycle works, ideas derived from watching others ride and from her own experiences riding tricycles. She knows to sit on the seat, to put her hands on the handlebars to steer, and to use her feet to move the pedals. When she first gets on a bicycle she tries to apply what she has observed, but soon realizes that there is more to it, something that can neither be taught explicitly nor be learned by observing others. The missing ingredient is balance. A child must learn to balance through trial and error, interpreting the feedback that the bicycle gives her in order to learn how to lean one way or the other. Eventually the child learns how to balance and ride the bicycle successfully, but learning does not stop. As a child gains experience riding a bicycle she refines her technique, learning such harder skills as going faster, crossing gravel, manipulating gears, riding with no hands, or ‘popping a wheelie’, all of which require a greater mastery of balance.

Now consider another learning process: learning how to use an automated telephone system. Most telephones in a business or home are connected to such a system, which adds functionality to the traditional telephone through features such as *call-waiting*, *voice mail*, and *call-transfer*. These features greatly extend the traditional telephone, and have been widely adopted. Business practice and personal needs encourage a user to learn the basic features. This learning is mediated by explicit instruction which lays out the patterns of key presses that access the system’s functionality. Explicit instruction is needed because it would take an impossibly long time by trial and error to discover that ‘*709’ accesses the voice mailbox or that hitting ‘*’ three times undoes the last action. But with the aid of documentation and help menus, users gradually add to their knowledge and can handle more features.

These two interfaces – the bicycle and the automated telephone system – share the property that, through use, users learn continuously, increasing both knowledge and ability, but they differ in how this ongoing learning occurs. With the bicycle, learning occurs implicitly. In contrast, learning the telephone system requires explicit, conscious thought, attending to instruction, and adding information to a mental model of the system.

Donald Norman [2, p. 22] notes an important flaw in the telephone system. An automated telephone system’s design is hidden from the user, who cannot see how inputs, i.e. key presses, affect the underlying system. The problem is poor *visibility* and *feedback*. In comparison, the bicycle’s design does not suffer from this problem as its input devices — two pedals, a seat, and handle bars — are visibly connected to the output devices — the two wheels and the frame’s position. This design makes it possible for the learner to observe the cause and effect relationships between them. In essence, the design of the bicycle allows it to *reveal how it works*.

This feature is called *self-revelation*. Self-revelation allows users to augment their knowledge incrementally[1] and implicitly. Through use, and sometimes by observing others[2], a user discerns cause and effect relationships, and discovers how things work. Self-revelation allows a new user to start using an application without prior instruction, and enables a user to continue her learning without explicit

[1]unlike the automated telephone system which is “all or nothing”

[2]A recent neuroscience study [3] found that people can subconsciously learn complex motor skills by watching others perform them.

instruction.

There is no doubt that self-revealing computer interfaces are highly desirable. Users would obviously prefer to sit down with a new application and use it immediately, and application developers would like to avoid the mundane task of creating support materials. Unfortunately, no one knows how to create self-revealing interfaces. There are only a few examples of self-revealing computer user interfaces, and no empirically tested techniques exist for making simple user interfaces self-revealing.

The original goal of this research was to discover principles governing self-revealing interfaces. As there is very limited empirical research on self-revealing interfaces, other self-revealing processes were examined. Language learning, a universal, self-revealing, and life-long learning process, became a focus of this research. Language and user interfaces have a lot in common: both are communication mediums involving humans: one human–human and the other human–computer. This research is not the first to notice and use the analogy between user interfaces and language (§3.1), but its approach differs in that it is concrete, and allows for experimental validation.

This thesis focuses on the *bootstrapping* phase of learning. Learning of languages or user interfaces has two distinguishable phases[3]: bootstrapping and refinement. Bootstrapping is the initial learning of the basic concepts. It is followed by a much longer, less intensive phase of ongoing learning, in which knowledge is added incrementally and skills are gradually refined. The bootstrap phase of mother-tongue language learning is self-revealing. Not knowing any language, a young child cannot be explicitly told how to learn language. So how does she learn? A child gradually builds an understanding of the language used in her immediate environment. She observes the linguistic interactions of others, normally parents and siblings, internalizing patterns of language usage and the effects of using language. She attempts to use language and observes the reactions, verbal or otherwise, of their communication partners. Early language learning occurs implicitly and naturally: it is a self-revealing process. Language refinement begins after children have grasped the basics of language, and continues throughout their lifetimes. It includes the gradual addition of vocabulary and grammatical constructions, which are acquired implicitly by observation and experimentation in conversation, reading, and writing. A much smaller amount is learned by explicit study, such as the study of a dictionary or grammatical text. Thus, the refinement phase consists of both self-revealing

[3]The distinction between these two phases is similar to the distinction made in the HCI literature between novice and expert users.

learning and learning through explicit study. The bootstrapping phase was chosen for study because it is:

- the most important phase in user interface learning,
- the hardest phase for users, and
- the most studied phase in language learning.

First, a case study was performed to look at similarities between early language learning and early user interface learning. The case study followed a user through the first ten hours of un-aided, exploratory use of an initially-unknown interface. The analysis looked for patterns in the accumulation of the user's knowledge and skills. The observations revealed many similarities between how users learn interfaces without explicit instruction and how children learn language. One important component found in children's language learning, called *motherese*, was not observed in the case study.

Motherese is a well-studied, universal feature of children's language learning [26] [30] [32]. It is a set of standard techniques that adults use to enhance their speech when conversing with young children. It helps to emphasize important features and guide children towards correct utterances. An experiment was thus designed to discover if users could benefit from motherese-style instruction in early user interface learning.

This thesis takes the distinguishing property of the bicycle interface — long-term learning-by-doing with minimal conscious problem-solving — to be the defining characteristic of a self-revealing interface. Research has examined the possibility that particular user interface elements, such as pie menus [4], may be self-revealing, but there has been no effort to discover how interface revelation[4] occurs in large-scale interfaces like word processors. The research project reported in this thesis is a study of the processes by which users learn large-scale self-revealing interfaces. This thesis presents the language-interface analogy, and the findings of a preliminary case study and subsequent experiment. These culminate in ideas for the future study of interface self-revelation, and a model for studying how novice users learn new interfaces through exploration.

Chapter 2 presents the motivations behind this research and previous HCI work related to it. Chapter 3 describes the analogy between user interfaces and languages, giving similarities between the two and describing motherese. A case study

[4]'Interface revelation' is the inclusion in the interface of presentations and affordances allowing long-term learning-by-doing with minimal conscious problem-solving.

of early, un-aided, exploratory learning of a complex user interface, in which similarities between the user's learning and a child learning language are observed, is presented in Chapter 4. An experiment evaluating motherese-like instruction for user interface learning is detailed in Chapter 5. Finally, the implications of this thesis are discussed in Chapter 6.

Chapter 2

Background

The aim of this research is to improve the learnability of user interfaces. A computer user becomes frustrated, and many computer users are intimidated, by the task of learning to use computer applications [5, p. 522] [6, p. 190]. He wants to be able to do things without explicit learning, which can only be done if interfaces are usable from the start, with further learning requiring minimal conscious effort. A user wants to learn to do increasingly complex things over time, without having to study. Therefore discovering how to design user interfaces that can be learned while being used is of extreme importance for the user interface community.

This chapter defines the context of the research presented in this thesis. Section 2.1 discusses the motivation for revisiting application learning. Sections 2.1.1 and 2.1.2 outline the foundation of these motivations in opinions and findings of the HCI community, and problems with current help techniques. The remainder of the chapter, Section 2.2, discusses self-revealing interfaces, which are an ideal for integrating user interfaces with help mechanisms. Section 2.2.1 describes self-revelation for entire, complex user interfaces. Sections 2.2.2 and 2.2.3 look at previous research on the self–revelation of interface elements, and entire interfaces, respectively.

2.1 Motivation

Learning is most effective when paired with a desire for understanding and applying what is learned. An application user, even a new one, knows what the application is used for: this knowledge directs use and learning. Rieman [6, p. 214] found that exploratory learning is almost entirely task-driven. He found that the typical computer user turned to help resources only when trial-and-error failed, absorbing details he had skimmed over in earlier browsing. Unfortunately, a help resource — such as a manual, tutorial, or online help — often presents information poorly or gives users more information than they need [5, p. 522].

When a user is able to start using a new application right away it is usually because common features of user interfaces are recognized from experience with other applications. Examples include consistency in menu structure, and in keyboard shortcuts for common operations such as cut and paste. This phenomenon, referred to as *learning transfer* or *transfer of training* [7, p. 264–270], is beneficial for users that have had similar experience with other programs.

Unfortunately learning transfer is not always possible. An inexperienced computer user is unable to reap its benefits, and consequently struggles with basic concepts and tasks [5, p. 522]. Furthermore, some applications have little in common with other applications. In these cases a user must learn new concepts, features and/or techniques in order to use the application.

To improve the learnability of user interfaces it is necessary that a user is able to do useful things from the start and build knowledge through use. This research focuses on online, implicit instruction in support of exploratory learning that is guided by the concept of *self-revelation*. Before introducing self-revelation in Section 2.2, a review of literature supporting it is given in Section 2.1.2 and problems with current resources for facilitating user interface learning are discussed in Section 2.1.1.

2.1.1 Directions for the design of help resources

Designing support for interface learning is daunting: users who vary in the prior knowledge that they bring to learning must be able to learn. Consider the best instruction possible: human instruction given on demand by an expert with complete knowledge of the application's workings, the user's knowledge, and the demands of the task. Such instruction is receptive and intelligent: it can perceive when the user needs help, can understand exactly what the user wants to do, and can show them how to do it, while taking into account the user's existing capabilities. Such instruction is infeasibly expensive; less expensive alternatives must be provided. This section describes HCI literature supporting the need for such instruction with its important qualities: online, user-guided, and aware of the user's goals.

In [5, ch. 13], Shneiderman and Plaisant describe the need for improved, integrated, online help. In particular, they discuss *ubiquitous online help* mechanisms for providing *universal usability* [5, p. 522–523]. They claim that *online help manuals* are not sufficient, offering too small an improvement over offline resources in the user's ability to comprehend and absorb what is taught.

Carroll [8] notes that new users are *active* and prefer to steer their learning of applications. Design principles based on this observation advocate letting the learner lead, integrating training with the system, and focusing on real user tasks and activities [9, p. 607]. Carroll provides experimental validation for two approaches that follow these ideals. The *Minimal Manual* condenses the typical manual, removing repetition and all content not related to doing things, to provide instruction that is geared towards real application usage. The second approach, the *Training Wheels Word Processor*, constrains the exploration space of novice users by disabling more advanced functionality. In a later, related publication [10], Carroll et. al give experimental evidence that user-driven exploratory learning that is guided by user-set goals and problems enhances overall user learning.

Many other experimental studies show the utility of online help that is based on the needs of individual users. In most studies, help is human-generated in real-time because of the difficulty of determining when users need help and what help they need. Earlier studies, such as those by Pollack [11] and by Coombs and Alty [12], required users to interact directly with an application expert. More recent studies [13] [14] [15] used a Wizard-of-Oz technique to mask the fact that the help was being generated by a human. In no case has it been possible to generate intelligent help automatically.

A well-known implementation of user-driven online help is the *Microsoft Office Assistant*, known as *Clippy*. Clippy provides online help to Microsoft Office users. In addition to other functionality, Clippy provides implicit help, inferring, correctly or incorrectly, user goals and alerting users when it has assistance related to them.

Unfortunately, by appearing when the user does not want help, Clippy is highly annoying. Carroll and Aaronson [13] similarly found that users are also annoyed when superfluous help was provided. They noted also that help must appear at the right time for it to be applicable and easily understood.

2.1.2 Problems with current interface learning supports

Interface learning is supported by many types of resources, including manuals, web sites, help databases, and tutorials. Experienced users know the properties of these

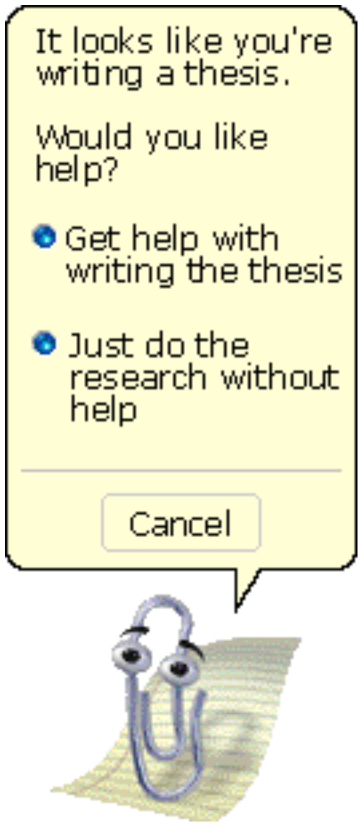

It looks like you're writing a thesis.
Would you like help?
Get help with writing the thesis
Just do the research without help
Cancel

Figure 2.1: The Microsoft Office Assistant, commonly known as *Clippy*, is based on *Microsoft Agent*, a platform for creating software character agents that provide enriched computer-user interaction and more natural online learning [16].

resources and use them accordingly [6, p. 215], but novice users lack even this knowledge. Much research has examined how these resources are used and how they can be improved [17] [18] [9, ch. 13]. This section discusses some general problems with traditional help.

Shneiderman and Plaisant state that: “Users need a presentation that shows the relationship between the metaphors and plans they know and the new ones” [5, p. 536]. However, most application learning aids are largely external to the applications they support. Paper manuals are completely non-computerized; online help web sites are displayed within a web browser; even help databases, which are accessed through an application’s help menu, give information in a ‘help’ window, which is separate from the main application window. Using such resources requires a context switch away from the application. Context switches are costly in time and mental effort, which makes it difficult to integrate what is being taught with how to apply it. Separating instruction from the task also makes it harder to control the quality and clarity of the instruction.

While online help resources, like help databases, can lessen the degree of this context switch, they are not preferred by users [6] and tend to be less effective [5, p. 523]. This could be caused by a number of factors.

1. Online documents take longer to comprehend, possibly because text printed on paper is more legible than that displayed on a monitor, possibly because a limited-size window can only show a small amount of legible text compared to a paper copy, which lacks rigid space constraints [6, p. 216]. They also have slower random access than books, with which users have much experience.

2. Online help is displayed in a separate window or in a virtual layer on the interface window. In either case, the provided help may cover-up important parts of the interface window. It also removes users' focus from the application, making it difficult to try out what is learned as the help and application are not simultaneously accessible [6, p. 216].

3. A user exploring an interface often understands the application's domain using terms that do not match the terminology of the interface. This mistmatch is referred to as the *vocabulary problem*. Most online help mechanisms assume that the user knows the terminology used in the interface and can use it when searching for help. A user who does not know correct terminology must rely on visual search, which is harder in computer-displayed text [6, p. 215–216].

4. Online help resources are not designed or written well.

Despite its problems, online help is more readily available than offline help and potentially has access to information about the user and their context [6, p. 216]. For online help to be effective, the problems mentioned above must be solved. Solving these problems can be accomplished by inferring user intent and using it to provide instruction within the interface window using non-textual, diagrammatic means — such as pictures and arrows — to relate instruction to interface components.

Online help better supports the preferred style of learning. Most users learn interfaces by exploration, supported by instruction from manuals or other users [6, p. 214]. Exploratory learning is preferred because users learn best in response to an immediate need. The average user tries to avoid pre-learning material for which he sees no immediate need. Effective online instruction enables learning without having to access external sources.

Interface learning should be online and occur naturally, incrementally, and within the context of the user interface [6, p. 190]. When user interface learning occurs in the environment of the interface, it is easier to comprehend. Furthermore, coordinating learning with actual use of the application makes learning less overwhelming: users do not need to store away information to be digested and tried later, and they need not context switch between instruction and interface. An interface that is useful from the start, like most computer games, and attains this ideal is *self-revealing*.

2.2 Self-Revealing Interfaces

An application with a *self-revealing* user interface is one which can be learned implicitly while using it. A Self-revealing user interface does not require explicit learning by the user: the interface presentation ensures that feedback and affordances are immediately obvious to the user; feedback given in response to actions provides users with the implicit information needed to comprehend their effects; the affordances require only actions that are easy and familiar, the user's ease and familiarity increasing with use. In essence, there's nothing to learn explicitly. As a user continues to use a self-revealing interface he continues to learn. This is similar to the life-long learning of a first language, which continues throughout life. The ideal of self-revelation should be a goal of all interfaces.

The use of text is a common feature of help resources. Reading written text requires full attention from the reader: it cannot be processed without explicit attention. Self-revealing instruction, in addition to being user-driven, must not intrude on the learner as textual instruction does. The learning resources discussed in Section 2.1.2 are therefore not elements of self-revealing interfaces: all are examples of explicit instruction.

A purely self-revealing interface would not need to provide users with additional help resources. But, while self-revelation can enable natural learning within an interface, learning in a self-revealing interface may not be as fast as the focused study of explicit support materials.

To date, most research on user interface learning has been focused on how to best create materials for explicit user interface learning in a given interface (e.g. [18], [17]). This focus is misguided. Instead, researchers should focus on *how to* *design user interfaces so that they reveal how they work through their use.*

2.2.1 Non-trivial self-revealing interfaces

Self-revelation can be applied to individual components of user interfaces: a user interface element, like a menu or form, may or may not be self-revealing depending on its properties and the level of expertise brought to the element by the user. Because of this dependence on previous user experience, claims that elements are self-revealing are not easily tested experimentally. Most prior research on self-revealing interfaces studied individual components of interfaces, such as the *pie-menu*, described below. There have also been claims that interfaces can be made self-revealing by following design guidelines [19], but no set of guidelines has been experimentally verified. This thesis empirically and specifically studies how large-scale interfaces can be made self-revealing.

As stated, self-revelation does not always apply to entire interfaces. No user can immediately grasp an entire non-trivial interface, such as a spreadsheet. Thus, the usual definition of self-revelation is elaborated to make it apply to complete user interfaces.

> *An interface, consisting of a set of components, is self-revealing if users can, from the beginning, do useful work, and if the interface supports learning-by-doing, in which users are able to extend and complete their knowledge while doing useful work, and without requiring explicit re-course to documentation.*

2.2.2 Self-revealing interface elements

Although there have been claims that user interface components, such as tool-tips, are self-revealing, there has been little experimental research to validate these claims. One component, the *pie menu* (see figure 2.2), has received research attention as a self-revealing design, and has been studied empirically.

In order to use a pie menu, a user needs to know only basic computer skills: how to move and select with a mouse and the concept of a hierarchical graphical menu (menus whose items are sub-menus or commands). A pie menu opens, with the press of a button, to show a number of labeled menu items arranged in a circle. Moving the mouse in any direction engages one of its entries. Most users know that

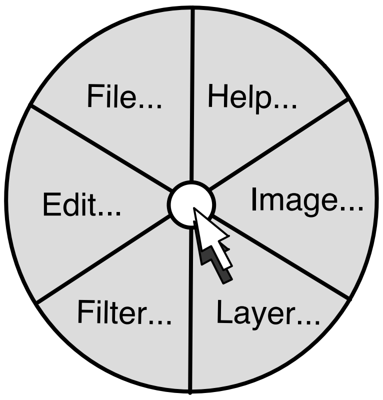


Figure 2.2: This figure shows a pie menu that has just been opened. A pie menus is a menu whose entries are pie shaped pieces arranged in a circle [4]. A central circular area can be used to exit the menu without making a selection. A pie menu pops-up with its centre beneath the cursor, as depicted. At this point, any mouse movement engages an entry, automatically opening sub-menu entries (a process referred to as *blossoming*). Releasing the mouse button selects the currently engaged entry, resulting in the execution of an action or the menu being closed.

pop-up menus open when a mouse button is pressed and that moving to a menu entry and releasing the mouse button selects it. Thus it is possible for users with moderate computer experience to immediately comprehend how to use pie menus without any explicit instruction.

Further learning of the pie menu consists of becoming faster at menu item selection. Selection of a particular menu item involves a consistent sequence of hand motions. This sequence is the same regardless of where the menu is invoked from. (This is contrary to docked menus, which require users to move the tracker to a specific location on the screen in order to invoke them.) Practice encodes the pairing, between a menu item and the mouse movement used to invoke it so well that users can often select successfully even before the menu is displayed. Experiments with pie menus [20] [21] substantiate its self-revealing properties, which are interpreted in terms of *muscle memory* or *motor schema* [7, p. 391], a very low-level effect that is potentiated by physical practice.

2.2.3 Are existing interfaces self-revealing?

There is very little research that looks at the self-revealing properties of entire user interfaces. This section briefly discusses one study that attempts to address self-revelation in the large, and contrasts the approach to studying this topic taken by its authors with that taken in this thesis.

Constantine and Lockwood [19] did a study on interface self-revelation in the large, claiming that application interfaces, even those that are ‘radically non-standard’, can be made completely self-revealing by incorporating explorability, predictability, and intrinsic guidance into their design. Their claim is strong: self-revelation does not depend on the overall structure of the interface, only on how individual elements are designed, and furthermore, that existing design methods properly applied produce self-revelation. Their instructive interaction techniques include such features as embedded prompts, templates, tool-tips, work-flow highlighting, and consistent structure. They explain how these interface elements tell the user how to use them, eliminating the need for explicit instruction, but their claims are not given an empirical foundation.

The goal of this thesis, like the goal of their study, is to study self-revelation in the large, but using an empirical method. Doing so requires a hypothesis, which is found in the analogy between language and the user interface. This analogy is described in Chapter 3, and examined empirically in Chapter 4 and Chapter 5.

Chapter 3

The Language-Interface Analogy

The user interface, like a language, is a communication medium. It allows two entities, namely a computer application and its user, to communicate, sharing information and making requests, commitments and plans. The same functions occur in human–human communication, with language as its medium, as is discussed in Section 3.2.2.

If the user interface is analogous to language, is learning a user interface also similar to learning a language? Language learning is appealing as it is a self-revealing process: it continues throughout a person's lifetime, and with little conscious problem-solving. It is also the best-studied self-revealing learning process. Thinking about the user interface as a language led to reasoning about learning in the user interface. In studying the self-revealing process of language learning one expects to discover ideas for facilitating self–revelation in the user interface learning process.

The focus in this research is on early learning. Both language and user interface learning can be divided into two consecutive phases: a *bootstrapping* phase followed by a *refinement* phase. In the bootstrapping phase, learners comprehend, through use, the basic concepts, or *building blocks* of the domain. In language these blocks are common words and how words are combined to form proper utterances. In user interfaces they are the core visual atoms used in displaying the state of documents — i.e. products, the basic tools used to manipulate them, and how to act on these tools to create and manipulate documents. The bootstrap phase was chosen because

- much is known about early language learning, and

- it comprises the learning of the fundamental aspects of language[1].

This chapter explores the analogy between language and the computer user interface. Section 3.1 reviews other studies of this analogy. Section 3.2 looks at important aspects of language, and Section 3.3 discusses their user interface counterparts. Section 3.4 narrows the focus to early language learning, which is the appropriate analogy for early exploratory user interface learning. Universal properties of this process govern the organization of the observations in the case study given in sectrefsect:study:observations. The last section describes *motherese*, a particularly interesting component of the early language learning process. The effectiveness of motherese-like behaviour in user interface learning is explored in the experiment presented in Chapter 5.

3.1 Previous References

Very little existing HCI research discusses the language-interface analogy. A quick search of the HCI literature turns up many references to language, but most of them are the result of the omni-presence of language in human life. Still, there are some papers, albeit dated ones, which discuss explicitly the relationship between the user interface and language, and many others which refer implicitly to important similarities. This section briefly reviews the most relevant of the papers containing explicit references.

In an early article looking at methods for enabling effective human-computer conversation, Foley and Wallace, [22], [9, p. 483–484], classify the principles as being either *language* based or *psychology* based. They describe human–computer communication in graphical user interfaces as a language that uses pictures and actions in place of words to facilitate conversation. They further divide this language into two separate languages:

- one is used for the *display* of the application's state and data, which is spoken by the user interface and interpreted by users to update their knowledge of the system, and
- the other is used for *action*, which is uttered by users through input devices and results in the application interpreting and performing the actions commanded by the user.

[1]It is also the easier of the two to study empirically in user interface learning.

They advocate a human–computer interface which is, where possible, controlled with human-like conversation that is natural, adhering to sentence structure, and also efficient.

Nickerson [23] considers whether human–computer interaction should be designed to resemble human–human interaction. He lists and discusses a number of important properties of human–human conversation and whether they would be effective in human-computer interaction. The features discussed include: bidirectionality, rules for transfer of control, intolerance of silence, sense of presence, history, and common world knowledge. He notes that computers do some things better than humans, such as repeating boring tasks or tolerating long unaccounted for periods without communication. And vice versa, humans do some things better than computers like learning and tolerating deviations from the norm. He claims that whether human–computer communication is like human–human communication is irrelevant: what matters is that communicating with the computer is pleasant for users and helps them achieve their goals. He concludes that human–computer interaction should be modeled, not as two human peers conversing, but as an intelligent tool serving human users. While he agrees that some features of human–human conversation are indeed desirable features for human–computer interaction, he feels that the differing abilities of the user and computer make other features a bad fit.

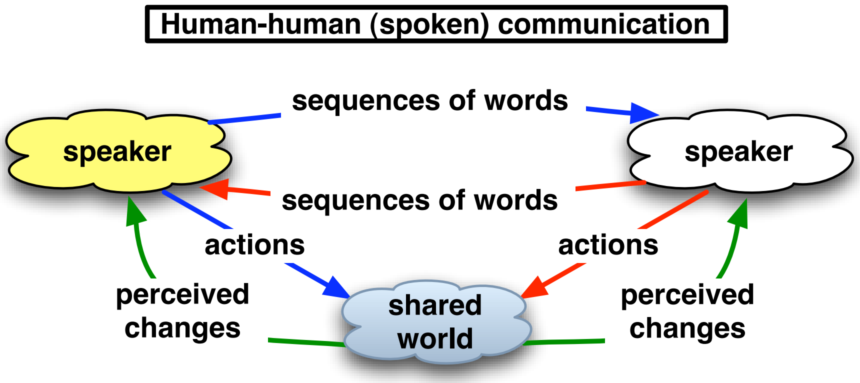
Rubinstein and Hersh extensively discuss computers and language in their 1984 book *The Human Factor* [24, ch. 6]. They discuss the relationship between language and computer systems at several different levels, programming a system at one extreme, and, at the other, facilitating the application-human dialogue by user interface command languages. In particular, they find that computer-human interaction should adhere to the user's expectations of how communication occurs, which is rooted in the user's experiences in human–human communication.

Richard Bolt's 1987 article about conversing with computers [25] discusses how human–computer interaction could be improved by having the computer evaluate a richer set of inputs in its conversations with users. He suggests allowing users to command a computer using *connected speech*, speech that is qualified by pointing gestures, facilitating interaction by speech and gesture recognition. Bolt claims that communication in user interfaces can be more expressive, or human-like, by adding more aspects of the rich communication found in natural human–human language. Nickerson [23] concurs, agreeing that human–computer conversation would benefit from the rich extra-lingual features of human–human conversation, which disambiguate and augment meaning in verbal communication.

All the aforementioned references highlight the similarities between human–

human language and human–computer communication and agree that at least some aspects of human–human communication enhance human–computer interactions. Most also suggest that making human–computer interaction more like human–human interaction is good, because it makes computer use more natural for users. The present research uses this analogy, but looks beyond their characteristics to specific features of learning. The aim is to gain ideas for enhancing user interface learning from early language learning.

3.2 Important Traits of Language



This section outlines important features of (spoken) language.

3.2.1 A communication and learning medium

A language allows two or more people to talk about things they know about, by providing generally understood representations, i.e. words, that stand for them. A word can represent an object (e.g. 'car'), an action on an object (e.g. 'driving'), describe an object (e.g. fast) or describe an action (e.g. 'quickly').

Language is essential for learning. It allows a speaker to introduce new ideas to others using description and generalization, including the use of metaphor and analogy. An important point is that learning language is inseparable from learning the world [26, p. 8]. Learning new words depends on learning what the words represent. This dependency is further described in Section 3.4.

3.2.2 Functional medium

Language is a communication medium with many uses.

1. Speakers can describe states of affairs: "The fat cat is under the hat."
2. Speakers can create and discuss abstract concepts: "1962 found the US and the USSR on the brink of nuclear war."
3. Less familiarly, perhaps, speakers can perform actions: "I promise to be back by noon."

The third use, the *performative use* of language, has an effect on its audience. Austin [27] coined the term *speech act* to refer to actions that language can perform. He shows that language can perform a variety of action types, such as asking questions, making commitments, giving orders, and making plans. Instrumental speech acts use language to demand things, while regulatory acts use language to control or command.

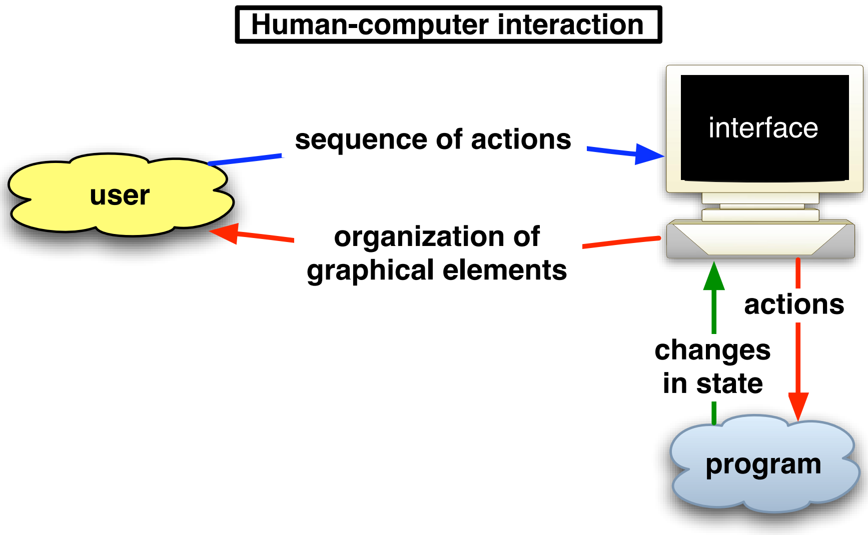
3.2.3 Context dependent

Language is heavily context dependent. Words often have multiple meanings, the meaning of a particular utterance (i.e. sequence of words) depending on the meanings of neighbouring words, and other aspects of the situation in which they are presented. Furthermore, all utterances are, in themselves, ambiguous, disambiguated by neighbouring utterances, and by the circumstances in which they are uttered [28, p. 207–219]. For example, the word "blue" can mean both the colour blue and the unhappy mood of an individual. The utterance "I'm glad," can mean that the individual is happy or that the person is named "Glad", or it might have another secret meaning. Thus one must take into account the surrounding context of utterances in order to understand their intention.

3.2.4 Discrete combinatorial system

Language is a discrete combinatorial system (DCS): a system with finite numbers of elements (words) and rules for combining them, that together produce an infinite number of valid combinations (utterances) [28, p. 84–85]. It is impossible to enumerate, let alone memorize, all the possible utterances that a language can create. Thus learning a language requires more than the rote memorization of given examples. A learner must internalize the language's rules for combining words in order to gain the language's full expressive power. Furthermore, in order to generate generally understood utterances speakers must also learn the domains in which these rules apply.

3.3 Interface Analogies



The properties of language given in the previous section have close analogies to properties of user interfaces. This section describes these analogies.

3.3.1 A communication and learning medium

Computer user interfaces allow users and computer programs to communicate about concepts in the domain of a program abstraction, which are presented by the interface. A vocabulary of graphical elements is used to represent objects and properties of objects in both the space of what the application can be used to do, e.g. documents the application can create, and the set of objects available to create and manipulate application objects, e.g. tools and commands. The interface presents a spatial display, built out of both types of these elements, that provides the user with a view of the application including the state of the user's creation, e.g. a document, and the state of the tools available for manipulating it, affordances and settings.

In order to learn an application, users must learn its interface. The language and images that are used in the interface's display are also used to explain what the application can do and how it works. Users must be able to understand these representations if they are to understand explanations that use them in describing the application's functionality. Some users learn the language of an interface from manuals and tutorials but the majority acquire it in use [6], and all users solidify their knowledge through practice, as speakers do with language. For example, the Edit menu groups together functions that support the rearranging of existing objects. Using editing tools evolves into automatic movement of the mouse towards the Edit menu.

3.3.2 Functional medium

User interfaces are media through which user-computer communication occurs, facilitating the control and monitoring of an underlying program by users. An interface, through graphical presentations, facilitates the creation and description of an application's tools, how they are used, and the objects they create and manipulate. This is analogous to the descriptive function of language. Users speak through interfaces to control applications by issuing commands that direct tasks and request data. This is similar to the functions of instrumental and regulatory speech acts, which use language to demand and control.

3.3.3 Context dependent

Interfaces are also intrinsically context dependent. An application's domain is the general context in which its interface is interpreted. The domain of word processors, for example, is formatted documents to be printed on paper.

Interpretation of interface components is dependent on the organization of elements in a visual display. The displays of components are perceptually grouped both spatially, by menus, toolbars, borders, vertical alignment, etc., and logically, by similarities and differences in their appearance and names. The correct interpretation of images and text used in an interface's visual presentations, interface utterances, depends on their immediate display context as well as the context provided by the application's domain.

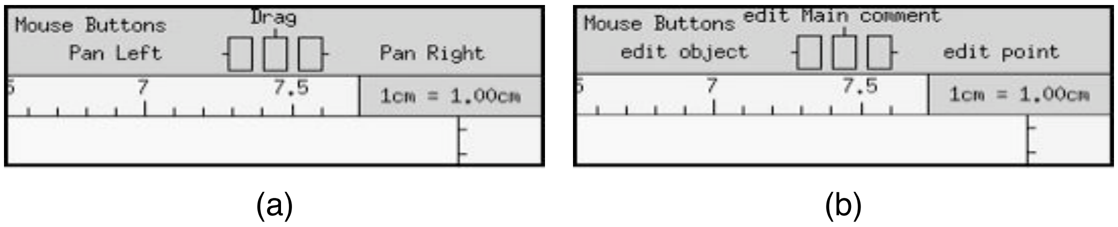


Figure 3.1: The actions performed by the mouse buttons in the XFig drawing program change depending on the context in which they are performed. In the program state depicted in (a) the mouse buttons can be used to change the document view. In the state depicted in (b) the buttons edit a selected object.

Sequences of user actions, user utterances, are also context dependent. Their consequences depend on the application's current state. This includes the selected tool, selected objects, selected documents, recent actions, and cursor position. Mode errors occur when users are mistake the context of actions. For example, textual input accidentally entered in command mode instead of insert mode in vi. Because interface actions are often ambiguous, it is very important that the application's current state be evident the user. Without feedback about the application state, a user finds it difficult, if not impossible, to use an interface, regardless of his or her level of understanding.

3.3.4 Discrete combinatorial system

User interfaces consist of two discrete combinatorial systems: one for command and one for description. The command DCS allows a user to combine a finite set of actions to achieve her unique goals. The description DCS combines a finite set of graphical display atoms to display the infinitely many possible states of an application. For example, a word processor allows users to create an infinite number of different documents through a combination of its finite set of actions, such as character and image input commands, which can in turn be displayed using a finite set of display elements, character and image displays. A user must learn how to combine actions to achieve her goals, e.g. create the documents she wants, and to understand interface feedback, e.g. the visual display of the document's current state.

3.3.5 Summary

In these ways the user interface is analogous to a language. In order for a user to use an application they must learn the language of the user interface. That is, she must learn how to communicate her requests to the application, through the interface, and understand the information communicated back to her, by the interface, about the current state of the application.

3.4 Early Language Learning

It is important to improve interface learning for users at all stages of the learning process, but, for the practical reasons mentioned at the start of this chapter, this project confines itself to early interface learning. For early interface learning — the first few hundred hours of use — the appropriate analogy is language learning in early childhood, on which there is a rich literature [29] [26].

Researchers in the field of children's language learning [30] [26] [31] [29] [28] have discovered many universal characteristics in children's acquisition of their native languages. These phenomena occur in children the world over, regardless of the language being learned. This section describes some of them. The observations of the case study (§4.3) find that analogous phenomena occur in user interface learning.

3.4.1 Production iteratively refines comprehension

Children learn new language concepts in a fixed order beginning by observing utterances. This observation begins the following process [26, ch. 6].

1. Children *notice*, in the speech of others, a word or pattern of words.
2. They *mimic* what they have noticed.
3. They *observe* the responses of others and changes in their environment.
4. They *update* their comprehension of its meaning and usage based on these observations.
5. They *try using it again*, possibly with alterations.
   $\longrightarrow$ step 3

6. They eventually comprehend the socially accepted meaning of the word or pattern and how to produce it.

Children iteratively learn about language through observation and use. At the earliest stage of language production mimicking is a simple, unaltered repetition of an example. Later, imitation evolves to include alteration based on the child's language understanding.

Researchers' opinions vary about how much imitation is done, but all agree that imitation plays an essential role in enabling incremental refinement of a child's language [26, p. 184–190]. Imitation allows a child to focus on a few aspects of language without fully understanding others. For example, children properly use the past tenses of verbs before knowing how to form them, considering a word and its ending to be a single unit. This is evident in the usage of proper past tenses of irregular verbs that are based on imitation, proceeding incorrect forms that are produced later when a child observes a general pattern in the creation of the past tense of a verb. For example, a child first imitates use of "ate", the past tense of "eat", but then learns the general rule for forming the past tense: past tense = verb root + "ed", and then begins forming the past tense as "eated", which is "eat" + "ed".

3.4.2 Comprehension and production refinement is selective

There is too much information provided in the speech examples that most children have access to. So, children only attend to what is interesting and salient [26, p. 179, 188–189]. Most children hear many more examples of the word "diaper" than they do "ball", but they will learn to comprehend and produce "ball" before "diaper" because they want to be able to refer to them, for example to ask to play ball.

Children also learn words to represent the things that they understand. This is because a child cannot associate words with concepts until she forms an understanding of what they represent. For example, a child starts to use the word 'gone' soon after she learns the concept of disappearance [26, p. 130].

3.4.3 Comprehension precedes production

Another aspect of children's language learning follows from the previous one: the set of spoken utterances that children can comprehend is a superset of the set they can produce to convey meaning [26, p. 140, 172]. Before a child begins speaking, she can discern subtle differences among sounds and exhibits consistent responses to particular words, both signs of language comprehension. A child responds to her name before she can say it. She can understand, and answer questions she cannot ask. This trend continues throughout a speaker's life. For example, production vocabularies, used in speaking and writing, are significantly smaller than comprehension vocabularies, used in listening and reading.

3.4.4 Vocabulary size, utterance length and utterance diversity increase concurrently

In early language learning, children go through a sequence of stages that show increases in utterance length, vocabulary size, and the complexity with which words are used and combined [26, p. 128–129]. A child begins producing utterances of single words, followed by two then three word utterances. She then jumps to creating utterances of arbitrary length. Concurrently, her vocabulary size increases, as does her mastery in combining words to form utterances. The speed at which children progress through these stages varies. The *mean length of an utterance* (MLU) is the average number of words spoken in combination. It has been found to be a better predictor of a child's linguistic ability than age, up to the point at which her MLU reaches 4.0 [26, p. 129].

3.4.5 Overgeneralization occurs transiently

When children are in the process of refining their comprehension of rules for combining and using words, they overgeneralize their application [26, p. 8]. Newly learned rules are used even when they do not apply, sometimes with surprising results. Consider the general rule for forming the past tense of a verb in English: add the suffix 'ed' to the root form of the verb. A child comprehends the rule by observing the pattern in the speech of others. She applies the rule to form the past tense for all verbs, even irregular ones, e.g. "mommy eated apple". With more experience a child comprehends exceptions to general production rules and handle their production differently.

A similar phenomenon occurs with vocabulary usage. A child starts by using unrelated terms to represent related concepts, goes on to make generalizations that are initially used too broadly, but are eventually refined to have the proper meaning. For example, a child may call her father's car "daddy car" and her grandfather's truck "bobba truck", without having an understanding of the general terms "car" or "truck". Later she may learn to use the general term "car" to represent all 4-wheeled passenger vehicles, including grandpa's truck. Eventually the child refines her understanding of the term "car" and learns the term "truck", which properly identifies vehicles like her grandpa's.

3.4.6 Comprehension and production refinement occur without instruction

It has been said, "If we taught children to speak, they'd never learn." [26, p. 93], meaning that children cannot learn language through explicit teaching [26, p. 5–6]. Instead, they learn by observing language in use [26, p. 154]. They learn a few things through explicit instruction, for example, the words for objects — parents handing them a ball while saying the word "ball" — but most language learning occurs implicitly, without instruction [32, p. 17].[2] Explicit instruction cannot explain how children learn to use words in combination to make meaning, or how they learn to use words like adjectives and articles [29]. The meaning of words and the patterns in their combination are extracted from the examples presented to them in their environment. From them, children implicitly develop the representations that allow them to make sense of and generate the language they observe [26, p. 93–94].

3.5 Motherese

Motherese[3] refers to a characteristically different style of language, used by adults when they converse with children[4]. Its characteristics include both qualitative differences in presentation, its use of pitch, tone and melody, and quantitative differences, in the structure of utterances and conversation flow. For over forty

[2]For example, up to eight years of age children add, on average, twenty-one words a day to their vocabularies [31].

[3]Motherese is also referred to as *parentese* and *child directed speech* (CDS) [32, p. 184].

[4]It shares many similarities with the way in which people talk to foreign-language learners, and deaf persons [32, p. 221]

years, it has been clear to language researchers that adult-child conversations are fundamentally different than adult-adult ones. Research [30] [26] [32] has shown that most of these differences are universal — present in cultures the world over, that children prefer motherese to normal speech, and that it plays an important role in children's language learning.

The following sections discuss important features of motherese.

3.5.1 Informative feedback

Parents use motherese in response to children's behaviour, even non-verbal behaviours like facial expressions and sneezes [26, p. 201]. Motherese gives children feedback about whether or not they are understood and gently corrects. The response to a child's utterance or behaviour is an interpreted version of it. The response is often repeated several times. This feedback provides gentle correction by example, exposing children to a (more) correct form of their utterances. It also lets them know if the adult has understood what they were trying to say. A child is informed of incorrect utterances either when the adult's re-phrased repetition does not match their intention, or when the adult responds with a clarifying question.

As language develops, a child's utterances show a progression of increasing linguistic abilities and the motherese they receive in return evolves to reflect it [26, p. 200]. It evolves from simple replacement correction, to correction that extends a child's partially correct utterance to a more completely formed version, by adding connectives and correcting conjugations, for example. Features like repetition become less frequent. For example, a baby's energetic pointing at an apple on the kitchen counter while making "aaghhh" sounds may result in their mother repeating by pointing to the apple and acknowledging by saying "A-PULLL. A-PULLL. Baby want A-PULLL??". In contrast, a more linguistically-developed child, pointing to the apple and saying "me appa", might receive the response "Would you LIKE an a-PULLL?"

3.5.2 Conversation flow and control

Children control what they learn in learning language [26, p. 94–95]. A child usually controls the direction of child–adult speech [26, p. 201]. Most verbal interactions are initiated by the child, with the adult's responses *acknowledging*, *repeating*, or *clarifying* what the child said [33]. In addition, the child tends to set the topic of conversation. Only about one-third of a child's speech in child–parent conversation is in response to what the parent has said, and over 80% of adult responses continue the child's topic [33]. These findings exemplify the self-revealing ideal that the learner should control the direction of her learning.

3.5.3 Simplified forms

Motherese speech has a simpler form than normal adult speech. It uses shorter sentences and simpler syntax [32, p. 17], avoids the use of first and second person pronouns, and focuses on the immediate environment — the here and now [26, p. 199]. This makes it easier to comprehend. While within the bounds of the child's knowledge of language, as noted in the next section, the language used in motherese is still complex, for example, containing a lot of questions [26, p. 200].

3.5.4 Sensitivity to learners' abilities

Motherese by definition "consists only of language material identifiable as primarily appropriate for speech to young children" [26, p. 196]. The content of adult speech to children changes as the child's abilities change [26, p. 200]. For example, early motherese uses repetition and not highlighting[5], whereas motherese in later stages uses highlighting, and little repetition. Adults also respond to children's non-verbal behaviour and ill-formed utterances, attempting to understand what the child is trying to communicate.

3.5.5 Phonetic differences

A teacher [34] noticed that her grade two students seemed to understand instructions better when she read them to them than when they read them to themselves. Why was this? Although they both contained the same wording, her reading of the instructions contained additional information not given by the written text. This information was conveyed through intonation, pitch, exaggeration, and other subtle features of her delivery that directed the students attention to the important points, all of which are observed features of motherese.

To accentuate key parts in speech directed at children, adults use "interpretable melodies: a rise-and-fall contour for approving, a set of sharp, staccato bursts for

[5]Highlighting is the use of phonetic differences to accentuate portions of utterances.

prohibiting, a rise pattern for directing attention, and smooth, low legato murmurs for comforting" [28, p. 279]; their speech is slower in tempo, higher in pitch, and exaggerated in its intonation [32, p. 15]. These features of motherese provide the child with extra information about an utterance's components and their relative importance. They accentuate word and phrase boundaries, distinguish different types of speech, and highlight information, all the while using well-formed utterances: far better formed than those of ordinary conversations [32, p. 16].

3.5.6 Summary

In essence, motherese is a form of implicit instruction that is given to language learners by adults. It responds to a child's speech, helping the child to identify important aspects of speech, and showing the child correct ways of communicating. Although it has not been proven to make learning easier [32, p. 81], its properties, universal presence in the language learning process [30, p. 129], the fact that children prefer it to normal speech [28, p. 279], and the large amount of attention it has received from the linguistic community support its importance in the language learning process.

This thesis applies characteristics of motherese to provide feedback to users in the user interface learning process. Chapter 5 describes an experiment that examines the effects of motherese-inspired feedback in guiding un-aided exploratory learning of a novel input technique.

Chapter 4

Case Study of Early Exploratory Interface Learning

As a user uses a self-revealing interface it implicitly provides the instruction needed to complement his normal exploratory behaviour. A user can start using applications with self-revealing interfaces without resorting to explicit instruction: ordinary exploration provides enough feedback. Thus, in order to start thinking about interface self-revelation one needs an in-depth understanding of a user's exploratory learning behaviour.

This chapter presents a case study of early exploratory user interface learning in which the similarities and differences in language and interface learning are explored. Case studies suffer from a variety of shortcomings, including lack of control and bias. Therefore, their results should be subjected to further experimental confirmation. This is done in Chapter 5.

4.1 Goal

The case study aimed to identify patterns in exploratory learning by excluding all help materials and input from knowledgeable helpers. Of particular interest was the changes in user exploration as a user's knowledge of an application and its interface increases.

Several studies [35] [36] [6] [37] examined how language used in user interfaces affects inferences users make about an interface during exploratory learning.

By contrast, the present study focused not on language use in the interface, but on resemblances between early language learning and early exploratory interface learning. Thus, instances of user learning analogous to observations of children's language learning were sought, to test the hypothesis that commonalities between language and the user interface extend to commonalities in learning.

4.2 Method

The case study looks at initial, un-aided exploratory learning of a complex application over a one week period. This section describes the interface, the user studied, and the learning task the user was given.

4.2.1 Study setup

The study used Adobe Photoshop version 6.0 running on Windows 2000. Photoshop is advanced two-dimensional image editing software. It was chosen because it is an example of a complex interface that includes a wide variety of interaction techniques and widgets plus a strong interface metaphor.

The case study took place during five two-hour sessions completed within a single week. Sessions were recorded in *.avi* files using Easy Video Capture screen capture software. The resulting log is an accurate, real-time transcript of mouse input and application feedback. The video transcript was supplemented with the user's recollections, captured in written notes immediately following each session. Transcript, notes, and recollections were, at the end of the learning process, written into a journal, to be analyzed together. Patterns observed while writing the journal were carefully authenticated using the video transcript. The result is an authentic, detailed, and intimate understanding of the user's actions.

4.2.2 The user

The user was the author of this thesis. This choice of user provided the deep accurate detail needed to notice and understand obscure parts of the user's learning process. The risk in this choice is bias and wishful thinking, which is mitigated, but not removed, by the transcript. The trade-off between descriptive intimacy and bias is characteristic of case studies, which explains why a case study is considered only one aspect of the research process. The first person singular will be used to refer to the case study user, while the the third person male will be used to refer to the general user.

I had minimal previous experience with Photoshop, despite some experience with similar image manipulation applications. I had rudimentary knowledge of layers, a core concept in Photoshop's image manipulation model, but I had never worked with them previously.

4.2.3 The learning task

I was given the task of combining facial features, such as noses, ears, eyes, mouths, eyebrows, or hair, from the portraits of four male professors to create a composite portrait. The composite was to contain at least one facial feature from each of the four individuals and to have consistent skin colouring. Two other restrictions were in place: I could use only low-level tools, such as cut & paste, paintbrushes, transformations, and manual colour adjustment, and I could not access any documentation, including online help, or ask for assistance. Three portraits were colour, one was black-and-white. They varied in lighting, geometry, scale, and camera position.

4.3 Observations

This section describes salient observations made from the records of the learning sessions. They are organized by the aspects of early language learning described in Section 3.4.

4.3.1 Production iteratively refines comprehension

Mimicking, copying observed behaviour, is the earliest overt sign of language learning in children. It occurs in user interface learning if users copy observed actions by rote. Sources of repeated actions might include actions of others, an example in a tutorial, actions tried previously, actions used in similar interfaces, or actions suggested by the interface presentation. Only the latter two sources for mimicking were available in the study. Mimicking in language learning is done to gain comprehension of an utterance's meaning, with responses from their conversational partners and resulting changes in their shared environment helping children to infer meaning. Similarly, presentation feedback resulting from user actions, showing changes in the states of tools and documents, helps an exploring user learn their effects. A user, like a child, makes initial inferences about the meaning of representations based on how they are presented and used.

Prior experience with another image manipulation program, GIMP, provided the user with incoming ideas about how to use many of Photoshop's tools. I had previously used tools with names and icons similar to Photoshop's selection, ink dropper, paintbrush and eraser tools. This incoming knowledge was used to form hypotheses about the effects of these tools, which were confirmed by visual feedback when I tried, i.e. mimicked, the usages in Photoshop. Such learning is often observed in user interfaces, and is called *learning transfer*.

I also learned several initially unfamiliar tools: the contrast & brightness tool and the layer ordering tool illustrate learning via exploration. I first tried the contrast & brightness tool while trying to adjust the colour in a layer of the composite image, which was taken from the black and white portrait. The layer ordering tool was discovered and tried while searching for a tool to support changing the position of a newly added layer relative to other layers. In both cases I first tried the tool, noting its name, icon, and physical location with respect to other tools. After trying obvious affordances, mimicking, and receiving feedback from the interface I was able to build up a general understanding of the tool, just as children develop understanding of mimicked utterances from the linguistic and environmental feedback they elicit.

Default settings for tools allow a user to gain an understanding of tool effects without understanding their full complexity. The graphical presentation of tool options and their effects allow a user to incrementally add to his knowledge, focusing on particular aspects of a tool in the context of default behaviour for other aspects. For example, I learned how to use the paintbrush tool with increasing skill over time, initially using it to paint basic coloured strokes, then trying tool options, another instance of copying, thus learning how to vary the stroke size, brush type, and paint opacity. The learning style is analogous to the child's iterative refinement of language comprehension through noticing, mimicking, and observing results.

Here and elsewhere, I learned about tools and how to use them by performing actions and observing the results. The process is analogous to children's language acquisition, in which production iteratively refines comprehension.

4.3.2 Comprehension and production refinement is selective

A child focuses on aspects of language that are relevant to his immediate needs and desires. My exploration, directed by the given task, sought tools to accomplish specific goals. I intentionally learned tools with effects that promised progress toward the goals and avoided others. I learned layer manipulation and colour adjustment tools because I needed to add, select and order layers, and to adjust colours in creating the composite image. I learned little or nothing about unneeded tools, like filter tools for warping images or tools for exporting images to different formats.

By the end of the study I user knew how to use many tools by invoking affordances through their visual presentations, but only three tools could be invoked using shortcuts: step-forward — which is similar to redo, step-backward — which is similar to undo, and magnification. Shortcuts need more practice to internalize than actions performed on graphical widgets, but are useful because they invoke tools faster. My shortcut learning was selective. Shortcuts were only learned for extensively used tools. Repeated use gave much exposure to the shortcuts, opportunities to practice them, and a reason for learning them. Similarly, a child learns and refines his production of utterances that are central to his interests.

Another, qualitatively different, example occurred when I employed the contrast & brightness and the layer ordering tools for the second time. Despite understanding their effects, I did not know how to invoke them: locations of the tools' in the interface had been forgotten. Presumably, on first use I understood their effects only after invoking them and observing their results. Thus, the connection between the need and invocation method was made only weakly. When the tool was rediscovered, the connection was made between what the tools did and how to use them, which was then available for subsequent use. This phenomena is akin to the time lapse observed between a child's initial understanding of concepts and the use of words to represent them. Like a child, I did not have a reason to know how to use the tools until I had seen their results.

4.3.3 Comprehension precedes production

A child understands language before he can speak. Once a child can speak he understand more than he can say. Vocabulary for comprehension of spoken language remains larger than for speech throughout life. Similarly, interface users must learn what an application can do and the actions needed to do it. Doing so requires a user to comprehend the interface's visual presentation and the effects of actions, the latter depending on the former. When exploring an application, a user must know what he wants to do and how the graphical state presented by the application's interface will change if he is successful. Thus he understands how the interface presents the effects of actions that he cannot yet produce. This phenomenon of comprehension proceeding production remains as a user, like a speaker, continues his or her learning.

I was able to locate tools during exploration using information gained from their names, icons, and context. Before trying them, I formed preconceptions of them based only on their presentations. I tried out the lasso tool when I needed to select a subset of the pixels in an image, understanding from real-world knowledge that lassos are used to round up animals. I tried the colour balance tool when I needed a tool to adjust the colour part of one image to match the colour in the composite, anticipating that it would balance the colours to match. The tools did what I wanted in both cases, and what the tools could do and the results of applying them were understood before I used them.

4.3.4 Vocabulary size, utterance length and utterance diversity increase concurrently

Children's understanding of how words can be combined, vocabulary size, and length of utterances steadily increase over time. Similarly my ability to combine interactions, length of continuous interaction sequences and the set of known actions increased with use.

Early in the study, I performed actions one at a time. I knew a number of different actions but I was unable to combine them into continuous sequences, pausing after each action to observe presentation changes and decide what to do next. This behaviour is analogous to the one-word stage of children's language learning.

With practice, I began to perform actions in combination. For example, both the ink dropper, which selects the paint colour, and the paintbrush tools are needed in order to add strokes of coloured pixels to an image. At first I used one, paused, then the other. With practice, I began to use them together fluidly, not needing to observe their individual effects. A similar pattern was observed with the navigator tool. Zooming often required subsequent resizing of the image window, done to allow me to attend to details in an image. At first these were two unconnected actions; later, they were chained together into a continuous sequence of actions, like a two-word utterance.

I also learned to further qualify my tool usage by setting options, like children learning to use adverbs to modify verbs. For example, I learned to use varied brush sizes for painting different sized areas, and to use lower paint opacities for blending than for colouring.

4.3.5 Overgeneralization occurs transiently

Errors due to overgeneralizations occur when a child's learned mappings from words or combinatorial rules to their meanings are too general. For example, all animals are "dogs" or all verb past tenses are formed by adding "ed" to the verb's root. Overgeneralization is an example of a context error: a correct rule is applied outside the context in which it applies. Similar context errors occur during user interface learning. Negative learning transfer is a common example of a context error: a rule that is correct in one context is applied in an inappropriate context. For example, an action learned in one interface, when applied in a new interface, may produce unexpected and undesirable results. Negative learning transfer causes learning in the new context to be difficult, because the rule producing the inappropriate action interferes with formation of a new one. It causes user errors based on incorrect models of the effects of actions, just as overgeneralizations occur as the result of a child's incorrect language models of word meanings or of how words are modified and combined.

As mentioned, my prior experiences helped me to figure out how to use a number of Photoshop's tools that were similar to those used in the GIMP, but there were some cases where this prior learning interfered. An observed instance of negative learning transfer was related to paint colour selection for painting. In the GIMP, one can select the paint colour *while* using a painting tool, like the paintbrush or airbrush tools. In Photoshop, paint colour selection is done with the ink dropper, a tool separate from the brush tools. Negative learning transfer caused me to make the same mistake repeatedly. I first selected the brush tool, then had to switch to the ink dropper tool to select the paint colour, requiring a second selection of the brush tool. Eventually, I became accustomed to the Photoshop technique, choosing, with a slight hesitation, the paint colour with the ink dropper before selecting the brush. The error is similar to ones that a child makes when he has a partial understanding of language. For example a general grammar rule, used without its exceptions, causes a child to make systematic errors when speaking. An adult corrects these errors when responding to the child, often asking him to repeat back the corrected form, but, he repeats the same mistake in his responses. He does hear the parent's correction, but fails to assimilate it because it does not conform to his incorrect model. In time, the child's model evolves, like that of a user, to produce proper behaviour.

Another instance of negative learning transfer was rooted in real-world knowledge, and affected my initial conception of the Image menu's contents. I had recently learned that images were made up of several layers, and so I took the term 'image' to refer to the whole and 'layer' a part. The thinking led to the belief that the Layer menu contained tools that manipulated layers while the Image menu contained tools that affected images. Because of this thinking, I did not open the Image menu in her exploration, and therefore could not locate several tools. Eventually I accidentally opened this menu and learned that it contained tools operating on the *image in the active layer*, and not the *entire image*. Like children making overgeneralizations based on incomplete understandings of aspects of language, learning users can make mistakes due to incorrect understandings of interface concepts.

4.3.6 Comprehension and production refinement occur without instruction

A child cannot be taught language explicitly. He learns its many complexities almost entirely through the implicit means of observation and trial and error. In the study, I had no explicit means of learning, and yet I was also able to learn a great number of things through self-guided exploration aided by incoming knowledge, the affordances of visual presentations, and visual feedback showing state changes. I tried tools, whose appearances, names, and relative locations corresponded to my current model of the interface until finding an appropriate one. My exploration provided examples of tool functions and information about tool organization in menus and toolbars. I used these examples to build up more general understandings of the individual tools and associations between them. Similarly, a child experiments with observed words and language patterns to form generalized understandings of them and their relationships to one another.

I was able to learn a great deal about how Photoshop works. For example, all transformation actions are ended by hitting <enter>. When I went to the Image menu to select a new transformation after performing a rotation, all menu items were disabled except for rotate. I then read a status message telling me to hit <enter> to end the rotation. When menu items were again found disabled following the second transformation, I tried hitting <enter>, which completed this type of transformation. After observing that multiple types of transformations were ended by hitting <enter>, the general rule — transformations are ended by hitting <enter> — was internalized.

I also built up general understandings of the tools grouped in Photoshop's menus, learning that tools in the Image menu operate on only the active layer, that its Adjust sub-menu contains tools that change the colours of the image in the active layer, and that its Transform sub-menu contains functions for applying transformations to the active layer's image. I also learned that paint tools use the currently selected colour, and that all colour selection is done by the ink dropper tool. In fact, all learning depended on observation of the appearance and behaviour of the interface, including information from graphical images, their affordances and layout, and the previews and effects of tools. I was able, like a child learning the complexities of language, to learn a great deal and form generalized understandings without explicit instruction.

4.4 Conclusions

The behaviour of a user in early exploratory learning of a complex application was studied to find behaviour analogous to the behaviour of a child learning his first language. These observations support the hypothesis that the two learning processes are analogous. The similarities observed include the following.

- A user gains knowledge about new tools by first mimicking actions and then building an understanding of them by iteratively observing their effects and using them. This process is analogous to the language learning process, which starts with mimicked utterances that are consequently understood through an iterative process of observation and use.

- A user's exploration is selective, with learning corresponding to immediate goals. Similarly, a child learns to express language related to his needs and desires.

- A user demonstrates a greater level of comprehension than production, allowing him to find the tools he needs by recognition. Correspondingly, a child demonstrates understanding of language before he can speak, which makes possible experimentation with language.

- The number of tools known to a user and the complexity of their combination increases as he gains experience. This parallels the development of a child's utterances over time, which increase in vocabulary, length, and grammatical complexity.

- A user makes mistakes based on an incorrect model of how tools work. My initial model caused me to use tools incorrectly and prevented me from trying out others. A child also makes errors based on an incorrect and incomplete understanding of language.

- A user, like a child, is able to learn through exploration using recognition and feedback, without having to be taught things explicitly.

The observed similarities follow the analogy described in Chapter 3 showing that learning of a graphical user interface and of language follow similar processes.

A very attractive aspect of language learning is that it continues throughout life, like the examples described in Chapter 1. This aspect was not considered in the case study, but the model provides a conceptual frame in which to better understand ongoing learning. Again, experimental work is critical, although it awaits creation of a method capable of examining processes that last for years.

The case study provides evidence that aspects of exploratory user interface learning are similar to children's language learning. The observed features of interface learning are not exhaustive, but there is ample evidence that knowledge about language learning extends to learning in the interface. Moreover, because the process of language learning is implicit, language learning will provide insights for making user interfaces self-revealing.

Children's language learning is a framework within which interface self-revelation can be understood and improved. This conclusion is supported by observations in a case study, which provides hypotheses for formal experimentation. The next section briefly suggests some lessons for interface design based on the observations made here. Other ideas based on the analogy are discussed in Section 6.2.2.

4.5 Lessons about Interface Learning

The observations show that the analogy between language and the user interface (Chapter 3) is supported by parallels between early language learning and early interface learning. Whether the similarities suggest that interfaces are learned as languages, or whether they reflect common use of a general learning capability cannot, at present, be determined. Regardless, it is obvious that the analogy provides a new and fruitful method for understanding the difficult topic of interface learning. The next chapter applies this method to devise a self-revealing support for exploratory learning based on the type of feedback given to children learning language. Before that, the remaining sections speculate briefly about immediate practical lessons for designing self-revealing interfaces.

4.5.1 Production iteratively refines comprehension

In interface design, consistency is important in the use of textual and iconic representations, and also in the interaction techniques used for tools. Prior knowledge in similar settings is a user-provided source for mimicking in learning. A user tries what he knows and has a hard time when his model must change to conform to new, contradictory situations. Care must take taken in designing interfaces to be consistent in order to enable positive learning transfer and avoid forcing a user to maintain inconsistent models, which is very difficult to do.

4.5.2 Comprehension and production refinement is selective

The direction of exploratory learning is determined by the user's immediate goals, not the imaginary goals set out in tutorials or manuals. The direction is also selective as a user learns what he is able to understand. An interface needs to provide online help mechanisms that understand user goals and support the learning needed to achieve them. An adaptive user interface tries to understand and adapt to the user's needs. It uses knowledge of the user's current skill level to modify the complexity of the interface. For example, it presents a subset of tools and functionality to a non-expert user to restrict the search space, making exploration easier. However, this adaptation is only a small part of what is needed. Adaptive user interfaces address the user's needs in terms of his general skill-level, not in terms of his immediate, individual needs. To support user learning well, help mechanisms must infer, or guess, the immediate intentions of individual users.

4.5.3 Comprehension precedes production

Graphical representations of tools, their affordances, and their relative organization allow exploring users to locate and use tools in the presence of unskillfully-defined goals. However, when user knowledge does not match up with the application's models, misconceptions prevent the user from finding what he needs. User interfaces must provide direct supports for exploratory search. This support must anticipate differences between user and application models, providing more open-minded assistance than online keyword search, which fails when a user's ideas about terminology differ from the application's.

4.5.4 Vocabulary size, utterance length and utterance diversity increase concurrently

As noted, graphical presentations of interface items are important because they allow a user to recognize more possible actions than he is able to recall. Exploratory learning leads to interface actions that work, but which are hard to combine with other actions. An interface should help a user move from individual actions to action sequences. The menu structure of DataDesk, which suggests follow on actions characteristic of a skilled statistician, is an existing example. Being shown how to combine tools encourages a DataDesk user to compose more complex utterances and to understand better the concepts they employ. Such suggestions and demonstrations give a user more complex examples to mimic than those given by graphical widgets alone.

4.5.5 Overgeneralization occurs transiently

An interface is usually designed so that action utterances can be formed from as small and consistent a set of rules as possible, yet many irregular forms, such as shortcuts, exist for efficiency or backwards-compatibility. A novice learns irregular forms by mimicking experts, who use them because they are efficient, but regular forms of activating the same features are also essential, for a user in the middle who is developing a general comprehension of the interface.

4.5.6 Comprehension and production refinement occur without instruction

Implicit learning relies on examples, feedback, correction, and generalization. Interfaces that respond to almost-correct input while showing correct input in their feedback help users to build models of the interface.

4.5.7 Summary

These examples show that the analogies developed in this chapter provide practical ways of making interfaces more self-revealing, lessons which are, once noticed, common sense. While useful, these do not amount to established methods for enabling self-revelation. The experiment in the following chapter takes us closer to this goal.

Chapter 5

Motherese Experiment

5.1 Overview

The case study of the previous chapter shows that many aspects of interface learning have close analogies to language learning. This chapter describes an experiment that explores a part of this analogy in detail: the analogy between motherese (§3.5) and interface feedback for novice users.

Of the aspects of early language learning discussed in Chapter 3, motherese is notably absent from the observations of Chapter 4. Thus, it is interesting to see if it has an effect on interface learning. In addition, it solves two problems inherent in experimentation on interface learning.

1. Choosing the correct instruction to give a learner depends on what the learner knows. Learners are extremely varied and the easiest way of standardizing prior knowledge is to chose an interaction of which all learners have no prior knowledge. Motherese is used to assist early language learners at a point where little or no knowledge of language exists.

2. Effective instruction depends on knowing the goals and intentions of learners. In experimentation, the experimenter can control user goals and expectations by assigning subjects a very simple task for which strategic planning is not possible. Motherese functions in an environment where the child's needs and desires are simple: the context of a child's utterances usually allow the speaker of motherese to know what the child is trying to express.

Thus normal experimental controls, together with a suitably simple and unfamiliar task, creates a learning environment in which interface features analogous to motherese will be maximally advantageous.

The experiment requires subjects to learn an interaction technique that is novel, using exploratory learning with no explicit instruction being given. A status bar and changes in the display provide basic feedback. One condition provides subjects with only this feedback; In the other condition subjects receive motherese-like feedback in addition to it. Observed superiority of the condition with motherese-like feedback would indicate that novice users obtain important knowledge from appropriate implicit feedback.

The remainder of this chapter describes the experiment and its rationale, concluding with a description of the experimental results.

5.2 Motherese

Motherese (§3.5) is used by adults when communicating with babies who are just starting to learn language. It is implicit by definition because it occurs at a point when the child is incapable of understanding explicit instruction. It is an almost universal aspect of children's exploratory language learning. The child attempts to mimic overheard speech, and an adult uses motherese to encourage, correct, and reinforce a child's utterances.

Novice interface users, faced with an unknown interface and no opportunity for explicit instruction are forced to learn by exploration. Graphical user interfaces provide affordances that are easy to identify and with which the user can easily experiment. An in language learning, motherese-like feedback should similarly encourage, correct, and reinforce the novice user's behaviour when trying to interact with an interface.

What kind of feedback in the user interface is analogous to motherese in language? Here is a possible example. In a graphical user interface a user activates a button widget by moving the tracker over it, followed by pressing a button on the mouse. When moving the mouse, a hand motion that aims the tracker at the centre of the button can be faster and more accurate than one aiming at a corner. It is easy to deduce that a user desires both speed and accuracy in this situation. Therefore, when the user clicks the button in the corner, motherese-style feedback could be used to move the tracker to the centre of the button before activating the button. Activating the button encourages and reinforces the user motion; moving the tracker to the centre gently and implicitly corrects it.

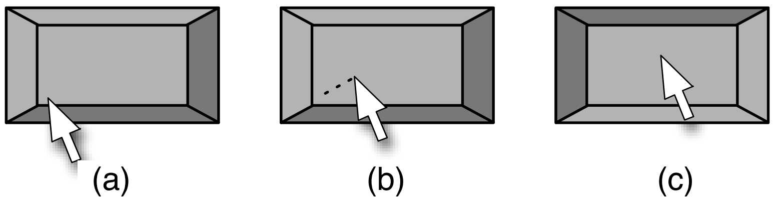


Figure 5.1: This figure illustrates the motherese instruction for a corrective button widget. (a): The user clicks in the lower-left corner of the button. (b): The motherese instruction animates the cursor from the click position to the button's centre. (c): After $\sim$ 500 ms the cursor reaches the button's centre and the button is activated.

This example shows that it is easy to devise motherese-style interface feedback when the goals of the user are known. Unfortunately, discovering the user's goal is, in general, a challenging, unsolved problem. Demonstrating by experimentation, in which the experimenter sets the users' goals, that motherese enhances exploratory learning provides added incentive for research on inferring the intentions of users from their actions.

5.2.1 Exploratory learning

In a field study, Rieman examined user interface learning in a natural situation [6]. He showed that exploratory learning is very common, but that users can rarely do it all by themselves: they require supporting reference material, like manuals, help pages or recourse to expert assistance. Furthermore, most exploratory learning is directly related to a goal or subgoal in the user task. These two observations support the above comments on motherese-style feedback. First, any instruction given should be unobtrusive and implicit, not attention demanding like reference to paper documentation or accessing a help system. Second, knowing the user's intentions is important for giving feedback that is directly related to the user's immediate goal.

5.3 Experiment

This section describes the experiment performed to investigate motherese-style interaction: the task to be learned, the experimental procedure, and the subjects.

5.3.1 Goal

The goal of the experiment is to examine the utility of properties of motherese in providing self-revealing user interface instruction. The experiment tests whether exploration aided by motherese-like instruction offers a learning improvement over traditional, un-aided exploration. The experiment does not compare motherese to other types of instruction as there are no other techniques for generating the self-revealing instruction with which to compare.

5.3.2 Task to be learned

The standard GUI interaction paradigms are well known. Therefore it is necessary to chose an obscure interaction technique to teach subjects in the experiment. At the same time, it is highly desirable to take advantage of the high level of expertise that exists for mouse use. Therefore, *mouse chording*, a mouse-based interaction technique with which very few users are familiar, was chosen. Mouse chording requires the user to press two or more mouse buttons simultaneously.

The task requires subjects to learn how to *transfer* colour from one simple rectangular shape to another using two different mouse chords. A transfer interaction consists of 3 steps:

1. A chord is pressed with the mouse tracker over a filled shape to *acquire* its colour.
2. The tracker is moved to an unfilled target shape, whose border colour matches the acquired colour.
3. A second, and different, chord is pressed, with the mouse tracker over the target to *drop* the colour onto it, filling it with colour and completing the transfer.

On each trial subjects had to fill a pattern of 3 targets using this technique. (See Figure 5.3 for an example pattern.)

Subjects were highly skilled in moving the mouse and clicking its buttons individually, but assumed to be unfamiliar with chording. Subjects were given no initial instruction on mouse chords. They were expected to discover through exploration (1) that chords were needed, (2) the chords that were active and where, and (3) how to use them to perform a transfer.

Each subject performed 60 trials, which were divided into 2 blocks of 30. Subjects performed one block with motherese-style feedback, and one block with ordinary graphical feedback. The transfer interaction technique was the same in both blocks, but the active chords (see Figure 5.2) differed. Half the subjects learned first with motherese and later without; the other half had this order reversed. The first block allowed for the observation of subjects learning both chording and the active chords, the second only which chords were active.

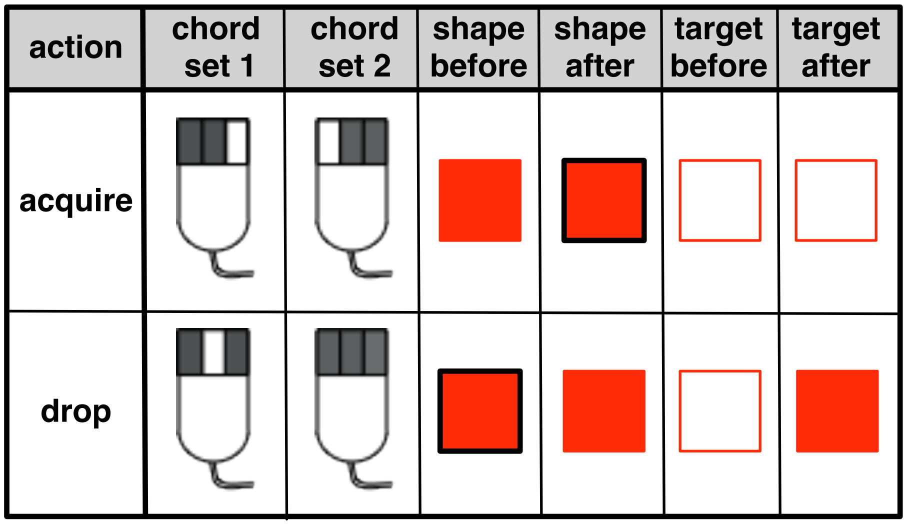

action
chord set 1
chord set 2
shape before
shape after
target before
target after
acquire
drop

Figure 5.2: This table shows examples of the effects of the *acquire* and *drop* actions and the two sets of chords used to activate them. Subjects had to learn to use one of the sets of chords in the first block and the other in the second, with this order counterbalanced across subjects. The *acquire* action acquires the colour of an upper shape when the active acquire chord is performed over it. The shape whose colour is currently acquired appears with a black border surrounding it, as depicted in the *shape after* column. Only one colour can be acquired at a time; acquiring the colour from a new shape unacquires the current colour and acquires the new one. The *drop* action successfully transfers colour to a lower target shape when (1) the active drop chord is performed on it, (2) it is currently unfilled, and (3) its border colour matches the colour that is currently acquired. A successful *drop* action results in a target being filled, as depicted in the *target after* column, and the acquired upper shape being unacquired, as depicted in the *shape after* column.

A fixed set of 30 target patterns was used. The order in which the patterns were presented was the same for each of a subject's blocks, but was randomized between subjects. This randomization was done to counter any effect that the different patterns could have on the results. For similar reasons, the presentation order for the two sets of active chords was counter-balanced. Because of these precautions, the choices of patterns and chord sets were not expected to affect the experiment results.

5.3.3 The experimental procedure

After initial instruction, subjects did 2 blocks of 30 trials each, separated by a short break. At the beginning of each trial the display showed four coloured rectangles in the upper half of the screen and three unfilled rectangles with coloured outlines in the bottom half of the screen. The subject's task was to transfer colour from the upper rectangles into each of the three lower target rectangles. This task was accomplished by acquiring colour from an upper rectangle using the appropriate mouse chord, moving the tracker to an unfilled lower rectangle, and clicking on it with a different chord. Colour was transferred, filling in the lower rectangle with the acquired colour, when the colour (and shape) of the lower and upper rectangles matched. The trial was complete when all three of the lower rectangles were filled. If all three transfers were not accomplished within thirty seconds, not including time during which motherese is presented, the trial was aborted. Following trial completion or timeout there is a short (3–5 seconds) pause, after which the next trial is started. The exception to this is the 30th trial in each block which is followed by either the inter-block break (block 1) or the end of the experiment (block 2). The subject was permitted to perform the transfers in any order, but the technique requires that a transfer be carried to completion before the next is started.

When the subject had completed two blocks, one with motherese and one without, the experiment ended. The entire experiment normally took about twenty minutes to complete.

5.3.4 Feedback

Traditional feedback

In both instruction conditions, subjects could see when the colour from a upper rectangle had been successfully acquired by the appearance of a black border around it, and that a drop, i.e. transfer, was complete by the lower rectangle being filled with colour. A status line — see Figure 5.3, at the bottom of the screen also informed subjects about the progress of the trial[1].

Corrective motherese feedback

Corrective motherese-style feedback follows erroneous button presses with a short animation depicting a correct action. For example, when subjects perform a chord other than the active acquire chord over an upper rectangle a small animation of the mouse buttons for the acquire chord being pressed over it is shown, as depicted in Figure 5.4. The animation first shows an image of the mouse with no buttons pressed, followed by images showing the press and release of the buttons for the correct chord, and completes with a selection border appearing around the rectangle. Unpressed buttons are shown in outline; pressed buttons are filled with black, as shown in Figure 5.2. Similar feedback, as depicted in Figure 5.5, is provided when errors are made in attempting a drop action. Corrective feedback is given when either an inactive chord is pressed, or an active chord is performed without resulting in an acquire or drop. The latter occurs if an acquire chord is performed somewhere other than over an upper shape, or if a drop chord is performed somewhere other than over a lower shape, or over a lower shape that is already filled or with an incorrect colour. This feedback is called *corrective* because it corrects a faulty utterance and encourages a correct technique.

Reinforcement motherese feedback

Once a subject is using chords to perform transfers, *reinforcement* motherese feedback is given. Reinforcement feedback, as depicted in Figure 5.6, consists of showing an animation of the entire interaction sequence involved in a transfer. The mouse icon appears pointing to an upper rectangle, its buttons are pressed and released followed by a border appearing around the acquired shape. The mouse icon then moves in a straight line to the lower rectangle, where the mouse buttons are again pressed and released resulting in the colour transfer. The acquire, movement, and drop are animated at the speed the subject took for each of these phases of the transfer just completed, and subjects are required to watch it to completion before beginning the next transfer. Reinforcement motherese was expected to improve facility with the mouse chords and the mouse motion by showing a hesitation-free version of the successful transfer. Maintaining the same speed gave the subjects an incentive to speed up.

[1]The status line was *not* pointed out to subjects.

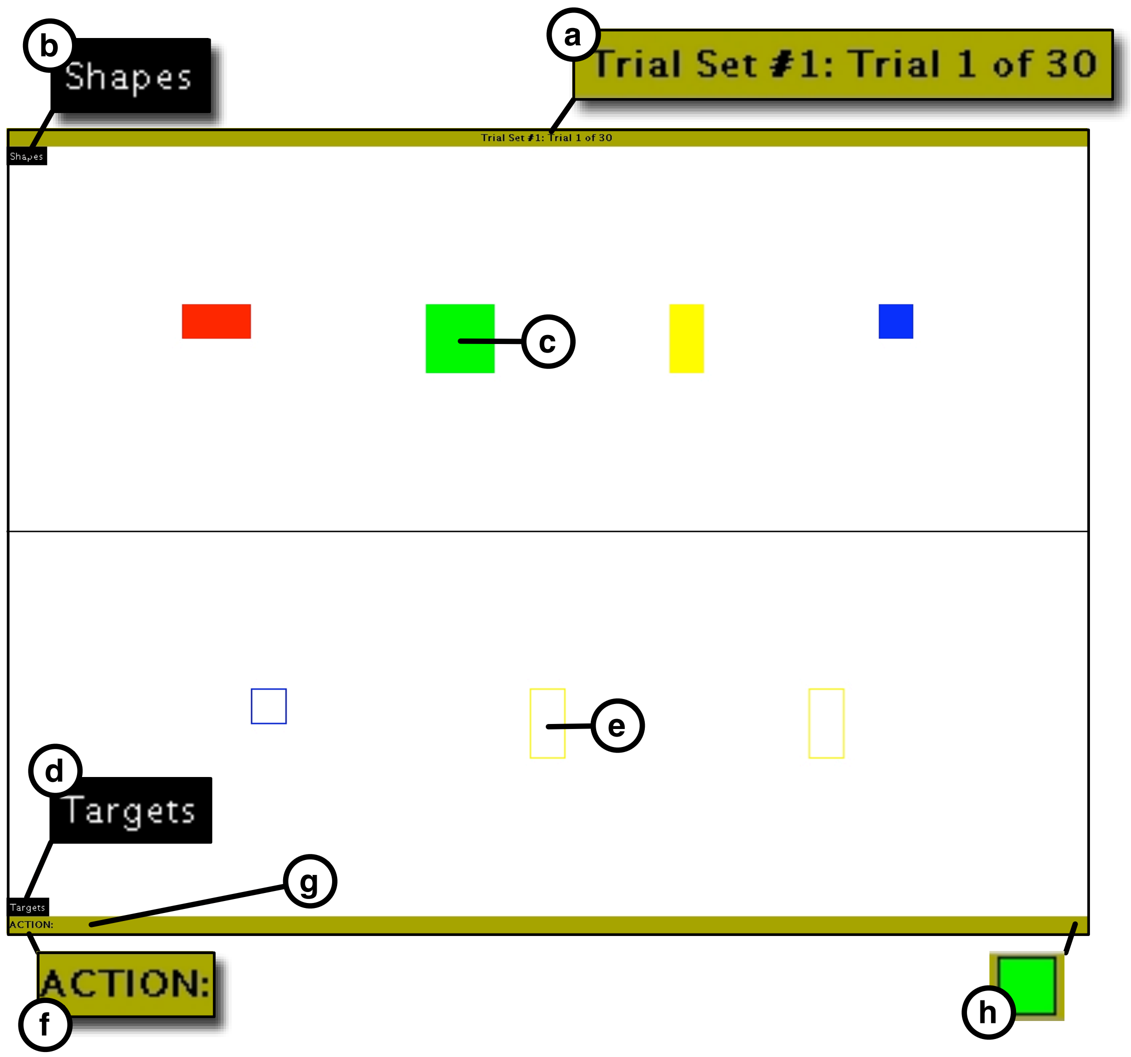


Figure 5.3: This figure shows a screenshot of the experiment interface with the areas of importance labeled, some of which are magnified for ease of reading. (a) identifies the location of information about the current trial number and set (i.e. block), showing the display for the first trial in the first block. Labels (b)–(e) point out parts of the large rectangular area in which the task is performed. It is divided into upper and lower areas by a grey horizontal line. (b) points out the ‘Shapes’ label that identifies (using the term used in the instructions, see Appendix C) the filled rectangular shapes in the upper half of the area whose colours are acquired. (c) points to one of the four *shapes*. Similarly, (d) shows the ‘Target’ label identifying the unfilled rectangular shapes in the lower area, into which colour is transferred, and (e) points to one of the three *targets* in a target pattern. (The target pattern shown is one of the 30 used in the experiment.) (f)–(h) highlight parts of the status line. (f) is a label describing (g), the location where, following the completion of a chord, a textual description of the action corresponding to it is briefly displayed. (h) shows where an image of the currently acquired shape is shown.

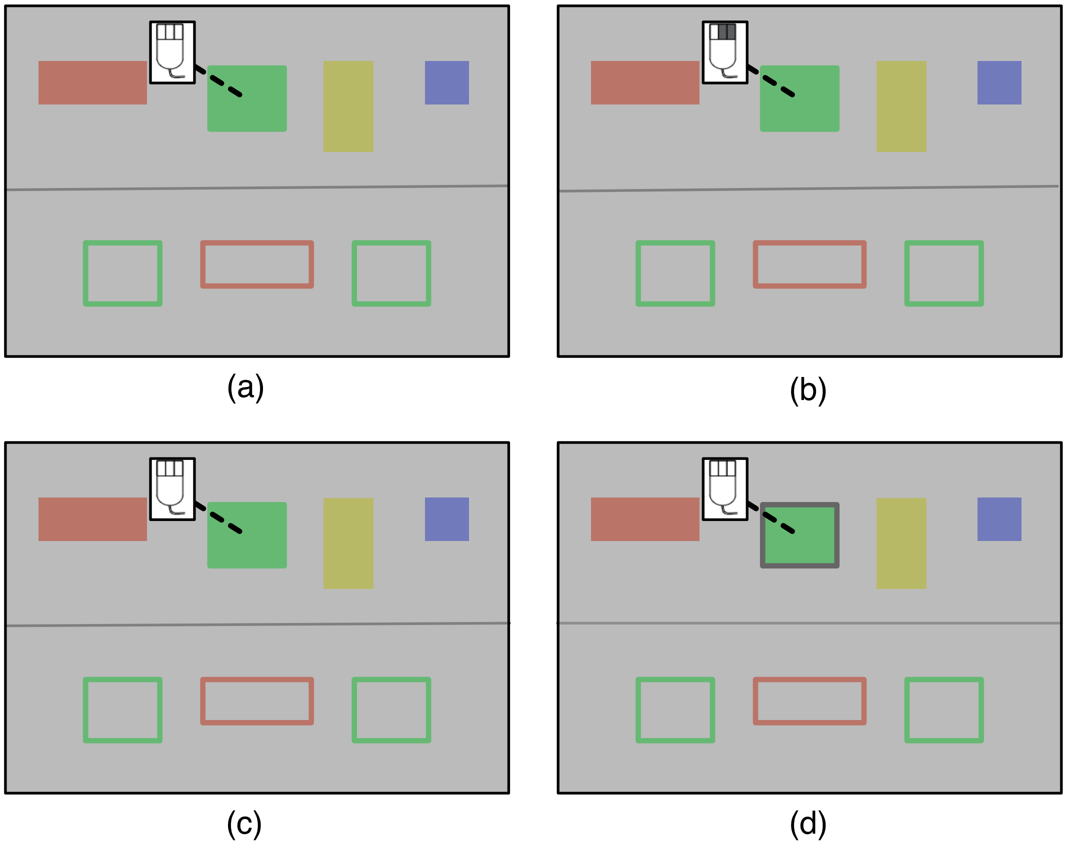


Figure 5.4: This figure depicts the correctional motherese-style instruction used in the experiment to show subjects how to *acquire* colour from an upper shape, in this case the green square. (a): The animation veils the task area and shows an image of a mouse with no buttons pressed. The image is connected by a dotted line to the centre of the square illustrating that the mouse is used over the shape. (b): 500 ms later the mouse image changes to an image with the middle and outer buttons darkened, showing that these buttons are pressed together. (c): After 1000 ms the image returns to a no button-down image, signifying the release of the buttons. (d): 200 ms later a border appears around the square, showing its selection (i.e. the effect of the action depicted). After 300 ms, the application returns to its previous state: the mouse image, line, veil, and selection border are removed. Subjects can interrupt correctional motherese at any time by starting a new chord. Doing so returns the application immediately. Chords started during correctional motherese that do not cause an action to be performed are considered a special case and do not result in correctional motherese. This is done to prevent a continuous cycle of motherese when erroneous chords are repeatedly pressed during correctional motherese.

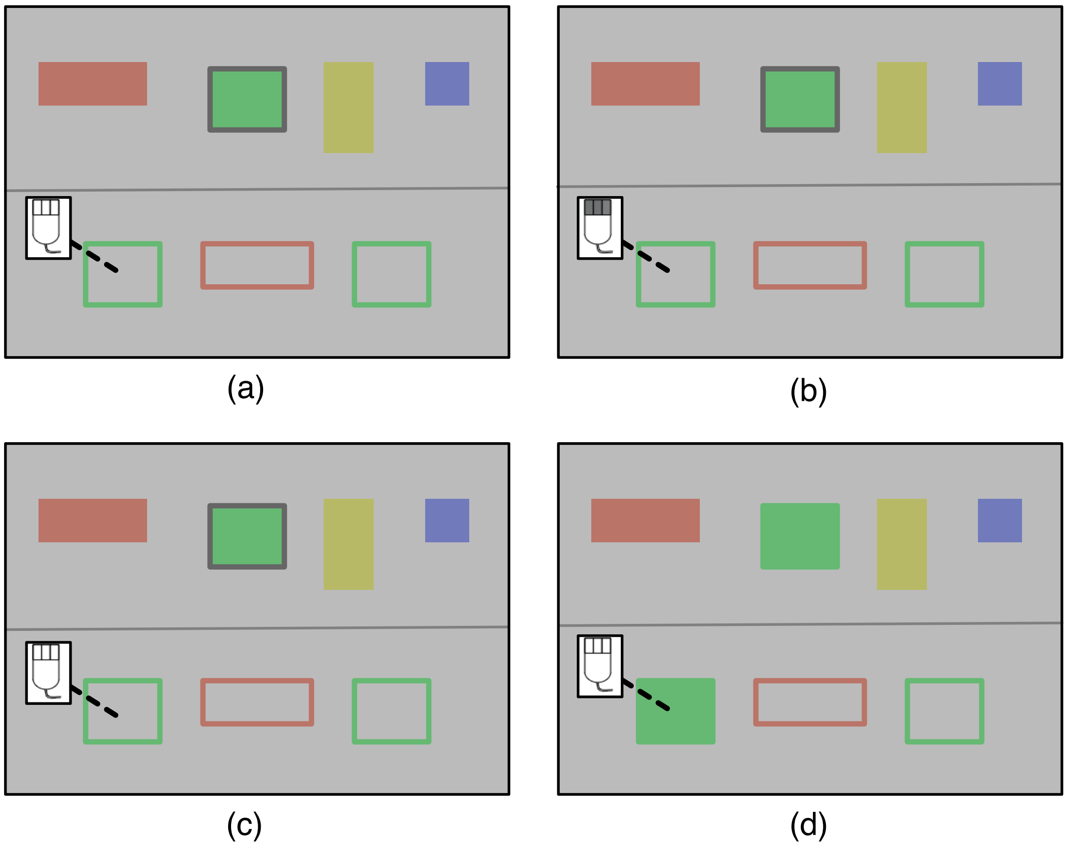


Figure 5.5: This figure depicts the correctional motherese-style instruction used to show subjects how to *drop* the acquired colour and fill a lower target shape, in this case the green square. (a): The animation veils the task area and shows an image of a mouse with no buttons pressed. It is connected to the square by a line to illustrate that the mouse is used over the shape. (b): 500 ms later the mouse image changes to an image with all three buttons darkened, showing that these buttons are pressed together. (c): After 1000 ms the image returns to a no button-down image signifying the release of the buttons. (d): 200 ms later the square is filled with colour illustrating the effect of the depicted action. The border on the upper green square is also removed to show its de-selection. After 300 ms, the application is returned to its previous state: the mouse image, line, veil, and fill are removed, and the selection border reappears.

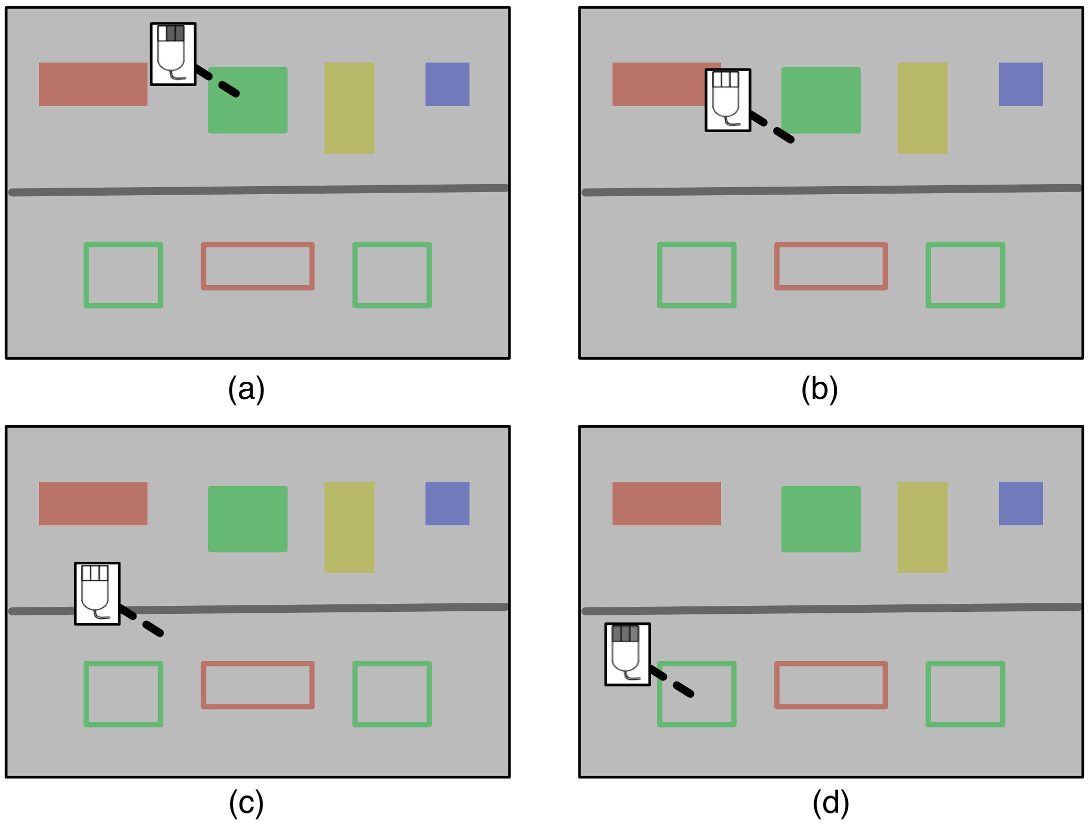


Figure 5.6: This figure depicts the reinforcement motherese-style instruction used to reinforce the transfer technique. This type of instruction is given following each transfer (in the motherese condition only). Each phase of the transfer (acquire, motion and drop) are played in the time the subject took to perform it. (a): The animation veils the task area and shows images that animate the playing of the acquire chord over the middle of the rectangle whose colour was acquired. (b) & (c): The image of the mouse (with no buttons pressed) is animated along the straight line to the centre of the target shape, highlighting the shortest path between the two rectangles. (d): Images are shown that animate the playing of the drop chord over the target rectangle. After 200 ms the veil is removed. Subjects must wait for the instruction to complete before they can begin the next transfer. This is done to encourage subjects to attend to it.

5.3.5 Apparatus

The experiment was run on a computer with a I GHz Pentium III processor and 512 MB of RAM. It ran Debian Linux, kernel release 2.4.24. A single Xterm was present on the display. The mouse was a true three button mouse, all buttons identical in size, type and shape. It had no scroll-wheel.

The experiment was implemented using Java, Blackdown version 1.4 beta, with the Swing GUI classes. Graphics and animations were created using the Java2D graphics API.

Experimental data for each subject was logged in two files: one containing all subject and experiment details, plus the timing and data of all events (whether subject generated or application generated), the other containing a condensed version of the same data.

The timing for all subject generated events was based on the time stamps provided by Java. Timeouts were provided by a Swing timer object. Timing for animations used a low priority thread that slept for short periods based on system time between updates.

5.3.6 The subjects

The subject group consisted of 13 computer science graduate students, 10 male and 3 female, all of whom were taking a graduate computer course on user interfaces and were naive as to the experimental hypotheses. Each subject had extensive experience using mouse and keyboard based interfaces for multiple operating systems and applications. In particular, each had extensive experience with the three button mouse. As a highly experienced group of subjects, they would be expected to benefit less than ordinary users from the presence of instructional feedback. Thus, strong positive results from this group of subjects is more convincing than a similar response from less experienced subjects.

The experiment was reviewed and received clearance from the University of Waterloo's Office of Research Ethics. Subjects were informed of the experiment procedure prior to participation, gave consent agreeing to participate, and were debriefed afterwards as to the study's purpose. Appendix B contains the information, consent, and debriefing materials that were used in the experiment.

5.3.7 Instruction

Subjects were given instruction about the experiment's format and their task. At the start of the experiment subjects received an overview of the experiment's structure: the number of blocks and trials, and the breaks between them. Task instruction was given by the application at the start of each block, and a hard copy was available to subjects throughout the experiment. Subjects were instructed that they were to learn a mouse-based technique and use it to act on three target shapes per trial, giving a description of shapes, targets, and their appearance changes resulting from performing actions. They were told to perform the task as quickly as possible. The instructions given are included in Appendix C.

5.4 Results

This section describes the experiment's hypotheses and empirical findings. Section 5.4.1 discusses the general method used in the analysis. Section 5.4.2 presents the hypotheses, and the statistical analyses are presented in Section 5.4.3.

5.4.1 Measurements

The analyses consider a dataset consisting of the results of 12 subjects, *S1–S12*. This dataset excludes the results of a 13th subject, S0, who was unable to learn the technique, timing out in every one of his 60 trials.

There are 60 results per subject, one for each of 30 trials in each of 2 blocks. 6 subjects had motherese instruction in the first block and no instruction in the second, the other 6 had this order reversed. This amounted to 30 trials for each subject in each of the 2 blocks.

The experiment has 60 cells, corresponding to the 60 combinations of trial and block. Measurements of 3 transfers for each of 6 subjects were collected for each cell, giving 18 transfers per cell, some of which were not complete. The data were recorded as average error rate and time on a per trial basis, as described in the *dependent variables* section.

Independent variables

The following is a description of the independent variables in the experiment.

- *Instruction Type* $\in \{motherese, control\}$ is the type of instruction given.
- *Trial #* $\in \{1..30\}$ is the trial number.
- *Block* $\in \{1, 2\}$ is the block.
- *Subject* $\in \{S1..S12\}$ is the subject identifier.

The *Instruction Type* is important for evaluating the relative effects of the instruction types. *Trial #* and *Block* relate the results to different points in the learning process, allowing learning to be examined over different periods.

Definition of transfer and trial completion

- A transfer is *successful* if it is executed to completion. Unsuccessful transfers are interrupted by a timeout.
- A trial is *complete* iff all 3 of its transfers are *successful*. An incomplete trial is interrupted by a timeout and has 2 or fewer successful transfers. A *partially complete* trial has 1 or 2 successful transfers.

Definition of transfer phases

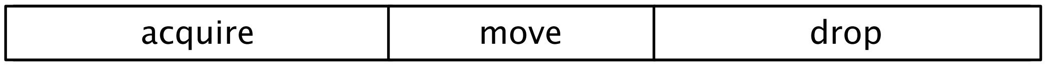


Figure 5.7: This figure shows the phases in a *successful* transfer. The *acquire* phase is the time spent acquiring the colour used in the transfer. (It is possible that multiple colours are acquired but only one the last one is transferred.) It starts with the first button press of the first chord following the start of a trial or inter-transfer period. The *motion* phase, represents the period, without chording, that immediately follows the acquire and in which the mouse moves to the target rectangle. The *drop* phase is from the end of the motion phase to the end of the transfer (i.e. the final mouse release of the drop action that transfers the colour).

A successful transfer consists of 3 consecutive phases as depicted in Figure 5.7.

1. The *acquire phase* is the initial part of a transfer spent acquiring the colour that is transferred.
2. The *motion phase* is the period, between the end of acquiring the colour and the beginning of an attempt to drop it. During the motion phase the mouse moves from the rectangle where the colour is acquired to a target.
3. The *drop phase* is the final part of a transfer in which the acquired colour is dropped, filling a target.

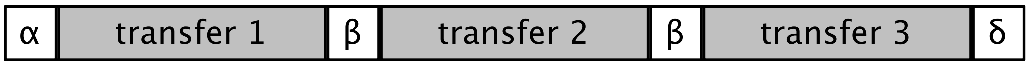


Figure 5.8: This figure depicts a *complete* trial. A complete trial consists of 3 transfer periods (shown in grey), surrounded by the non-transfer periods (shown in white) α, β, and, in motherese trials, δ. α is the time from the start of the trial to the start of the first transfer (i.e. the start of the first chord). In motherese trials, reinforcement motherese is given immediately following each transfer and is therefore contained in β and δ. The 2 β periods occur between transfers 1 & 2, and 2 & 3. They end when the first chord is started following the end of the reinforcement motherese (in motherese trials) or the last transfer (in non-motherese trials). δ occurs only in motherese trials. It starts following the completion of the 3rd transfer, and consists entirely of reinforcement motherese.

Definition of trial phases

The time taken by a trial consists of transfer time, for up to 3 transfers, plus non-transfer time. Figure 5.8 depicts the phases in a *complete* trial. (Transfer times are depicted in grey and non-transfer times in white.) Non-transfer time includes:

- α: the time from the beginning of the trial to the first button press of the first chord,
- β: the time between the transfers, which contain reinforcement motherese in the motherese condition, up to the first mouse button press of the first chord, following the reinforcement motherese in motherese trials, and
- δ: the time, which occurs in motherese trials only, from the end of the third transfer to the end of the trial, during which reinforcement motherese is given.

The first transfer starts with the first button press in the first chord following the beginning of the trial. The second and third transfers start with the first button press of the first chord following the end of the last transfer (in non-motherese trials) or the reinforcement for the last transfer (in motherese trials).

Dependent variables

- For each trial, let n_s be the number of successful transfers, $n_s \in \{0..3\}$.
- For each of the transfers $i \in \{1..3\}$ in a trial, let

$$a_i = \begin{cases} \textit{time for the acquire phase of transfer } i, & \textit{transfer } i \textit{ successful} \\ 0, & \textit{otherwise} \end{cases}$$
$$m_i = \begin{cases} \textit{time for the motion phase of transfer } i, & \textit{transfer } i \textit{ successful} \\ 0, & \textit{otherwise} \end{cases}$$
$$d_i = \begin{cases} \textit{time for the drop phase of transfer } i, & \textit{transfer } i \textit{ successful} \\ 0, & \textit{otherwise} \end{cases}$$
$$c_i = \begin{cases} \textit{the \# chords in transfer } i, & \textit{transfer } i \textit{ successful} \\ 0, & \textit{otherwise} \end{cases}$$

The dependent variables measured for each trial, which calculate averages across its successful transfers, are defined as follows.

- $\mathit{ClickTime} = \frac{1}{n_s}\sum_{i=1}^{3}(a_i + d_i)$
- $\mathit{MotionTime} = \frac{1}{n_s}\sum_{i=1}^{3}(m_i)$
- $\mathit{TransferTime} = ClickTime + MotionTime$
- $\mathit{Mischords} = \frac{1}{n_s}\sum_{i=1}^{3}(c_i - 2)$

These variables respectively measure the average time spent chording, time spent moving, time spent performing transfers, and number of unneeded chords. These measures are undefined in trials with no successful transfers.

Evaluation periods

The data was examined looking for evidence of both learning and refinement of the transfer technique. Evaluating learning requires looking at trials at the start of a block. Evaluating refinement requires a period of trials in which initial learning of the technique has completed for all subjects.

The *learning period* is used to evaluate initial learning of the technique. This period is defined to be the block of trials from the beginning of a block, trial 1, until the trial where the number of complete transfers stabilizes at 3.

The *post-learning period* is where subjects have learned the technique and are performing it with variability only in speed, *TransferTime*, and accuracy, *Mischords*. The start of this period is determined based on the performance of the subjects in both blocks to allow for comparison between blocks. This period continues until the end of an experiment block.

Outlier removal

Outliers were removed before starting each statistical analysis. Outlier removal was done using an iterative method [38], removing all data points lying outside two standard deviations of the mean iteratively until all points remaining lie within a range of two standard deviations of their new mean.

5.4.2 Hypotheses and statistical tests

The high-level working hypothesis is that *exploratory learning aided by motherese is more effective in speeding-up initial learning and refining already-learned actions* than un-aided exploratory learning. The following are specific measurable hypotheses organized by effects of the two types of motherese used.

Correctional motherese hypotheses

These hypotheses formalize the expected effects of the correctional motherese. H1 predicts a reduction in the bootstrap learning period length. H2 predicts improvement in clicking and movement times. H3 and H4 predict reductions in errors and time after initial learning is complete.

H1: With motherese learning is faster

Correctional motherese demonstrates how to click, guiding user exploration toward correct actions. Thus, subjects were expected to learn faster with motherese, producing fewer incomplete trials.

Speed of learning is evaluated by comparing the number of successful transfers in each trial, n_s, averaged across the subjects with and without motherese instruction. When this value stabilizes at 3 subjects are considered to have learned the technique. Subjects with motherese instruction are expected to learn the technique earlier in the trial sequence.

H2: With motherese transfer times on successful matches are expected to be lower and to improve faster

What subjects must learn is chording: that several buttons are pressed together, and which buttons to press. As all subjects were experienced mouse users, highly skilled at moving the mouse, better performance in performing transfers, *TransferTime*, was expected to be the result of better chording, *ClickTime*, and not the result of faster movement, *MotionTime*. Motherese shows users how to chord so subjects with motherese should speed up their chording faster than the control group.

To examine this, *TransferTime, ClickTime,* and *MotionTime* are compared in all trials with successful transfers, i.e. $n_s > 0$, and no mischords to see if (1) the time for transfers, *TransferTime*, decreases, (2) this improvement comes from a speed-up in clicking, *ClickTime*, and not a speed-up in moving, *MotionTime* — i.e. *MotionTime* remains about the same, and (3) this improvement is significantly larger for those with motherese instruction than without.

H3: With motherese accuracy is better in the post-learning phase

Correctional motherese provides a visual representation of the buttons used in the active chords. It also *re-tells* how to perform the action each time a user errs. These reminders are expected to cause subjects to internalize the actions better, resulting in fewer mischords after initial learning.

To test if motherese improves accuracy the average mischord rate for the two instruction types are compared. An ANOVA of *Mischords* by type of instruction in the post-learning period is performed to find the average rate for each instruction type and to check if they differ significantly.

H4: With motherese transfer times are faster in the post-learning phase

For the reasons described in H3, correctional motherese should improve task learning resulting in better performance. Because mouse movement is highly practiced, an improvement in clicking, *ClickTime*, is expected to dominate this learning effect.

This hypothesis is evaluated by comparing the average post-learning *ClickTime* in error-free trials, i.e. *Mischords* = 0 with motherese to that without. A one-way ANOVA of the effects of instruction type on *ClickTime* gives average times for each of the instruction type and shows if they are significantly different.

Reinforcement motherese hypotheses

The remaining hypotheses make predictions about how reinforcement motherese affects rates of improvement during the post-learning phase.

H5: With motherese accuracy improves faster in the post-learning phase Reinforcement motherese is expected to solidify responses in the user. If so, accuracy should improve faster in the presence of reinforcement motherese.

To test this hypothesis, the number of mischords, *Mischords*, is regressed against the trial number in the post-learning phase for each instruction type. The slopes of the regression for the two instruction types are compared to see if they differ.

H6: With motherese speed increases faster in the post-learning phase

Reinforcement motherese enables a user to judge his or her speed. It gives an incentive to improve by adding to the penalty for slowness. Thus, subjects might be expected to speed up quicker with reinforcement motherese than without.

This hypothesis is tested by regressing the transfer time, *TransferTime*, against the trial number, *Trial #*, in the error-free post-learning trials for each instruction type. The slopes of the regressions are compared to see if the rate of improvement for the two instruction types differ significantly.

5.4.3 Analyses

This section describes the findings of a statistical examination of the experiment results.

In the analysis, significance of the results is categorized as follows:

- *highly significant* (HS) if the probability of accepting the null hypothesis is $<$ 0.01 (1%),
- *significant* (S) if the probability of accepting the null hypothesis is ≥ 0.01 but less than < 0.05, and

- *not significant* (NS) otherwise.[2]

Before testing the hypotheses, one-way ANOVAs of the effects of *Instruction Type, Trial #,* and *Block* on *TransferTime* and *Mischords* were performed, for the trials with one or more successful transfers, without removing outliers. There were significant effects of all of these factors on *Mischords* and of all but *Instruction Type* on *TransferTime.*

Some of the analyses look at data for only the learning or post-learning periods (§5.4.1). The post-learning period was determined to be trials 16 through 30. An examination of both *TransferTime* and *Mischords* showed that by trial 16 the subjects[3] were consistently performing the technique, with little variability in either measure.

Correctional motherese hypothesis H1:

This experiment evaluates illustrative feedback that is activated during user learning and aims to guide users towards correct actions. H1 predicts that subjects with this motherese instruction learn quicker than those without.

To evaluate this hypothesis, the number of transfers successfully completed in each trial are compared by instruction type. This number is a value between 0 and 3 as each trial has 3 possible transfers to complete, any number of which can be successfully completed. Learning is therefore evaluated by the length of the learning period, which is the period before which the number of transfers completed stabilizes at 3. Figures 5.9 and 5.10 show graphs of these averages for each instruction type in blocks one and two respectively.

Figure 5.9 shows a graph of the successful transfer averages for each trial and instruction type in block one. The following observations about the performance of the two instruction groups can be made from it.

1. The motherese subject group averages 1.5 successful transfers, 50% of the possible transfers, in the 1st trial.

2. The control subject group remains at a level of 0 successful transfers from trial 0 to trial 7.

[2]These significance levels were set based on the overall number of tests performed, to make it highly unlikely that claims would be made based on false positive results.

[3]with the exception of *S5* in block one, where he or she did not learn the technique in the control instruction condition

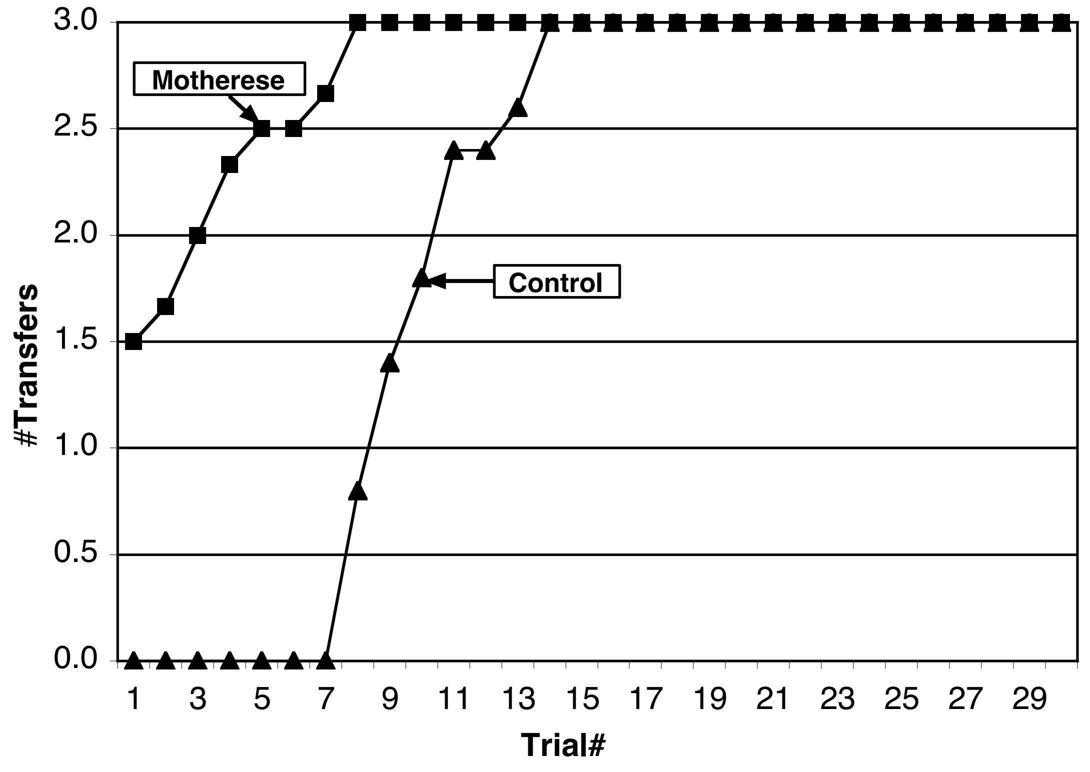


Figure 5.9: This graph shows the average number of transfers completed per trial for each type of instruction in block one.

3. The motherese subject group achieves 3 successful transfers for the first time in trial 8.

4. The control subject group achieves 3 successful transfers for the first time in trial 14.

5. The motherese subject group remains stable at the 3-successful-transfer level from trials 8 through 30.

6. The control subject group remains stable at the 3-successful-transfer level from trials 14 through 30.

7. The average for each subject group increases monotonically.

Observation 1 shows that some of the subjects with motherese were able to perform transfers successfully before the end of the 1st trial of the first block, and that this subject group as a whole was able to complete half of the possible transfers in this trial. No subject without motherese instruction was able to successfully perform any transfers until trial 8, at which point the entire motherese group was already completing all 3 transfers. Observations 3 and 5 together show that the learning period for the motherese group is about 8 trials. Similarly, observations 4 and 6 show that the control group's learning period is about 14 trials. The last observation shows that the average transfer rate increases with experience, both with and without motherese. These observations show a shorter learning period, by 6 trials, for the motherese instruction group. The first two observations show that subjects with motherese were able to perform a transfer within the first trial, i.e. 30 seconds, while no subject in the control group, whose subjects exploration received no instruction, managed to perform a transfer in any of the first 7 trials, for more than 3.5 minutes. This supports a strong bootstrapping effect of the motherese instruction in quickly guiding user exploration towards correct behaviours.

Figure 5.9 shows a graph of the successful transfer averages for each instruction type in the block two trials. The following observations about the performance of the two instruction groups can be made from it.

1. The motherese subject group averages 2.5 successful transfers (∼83% of the possible transfers completed) in the 1st trial.

2. The control subject group averages 1.5 successful transfers (50% of the possible transfers completed) in the 1st trial.

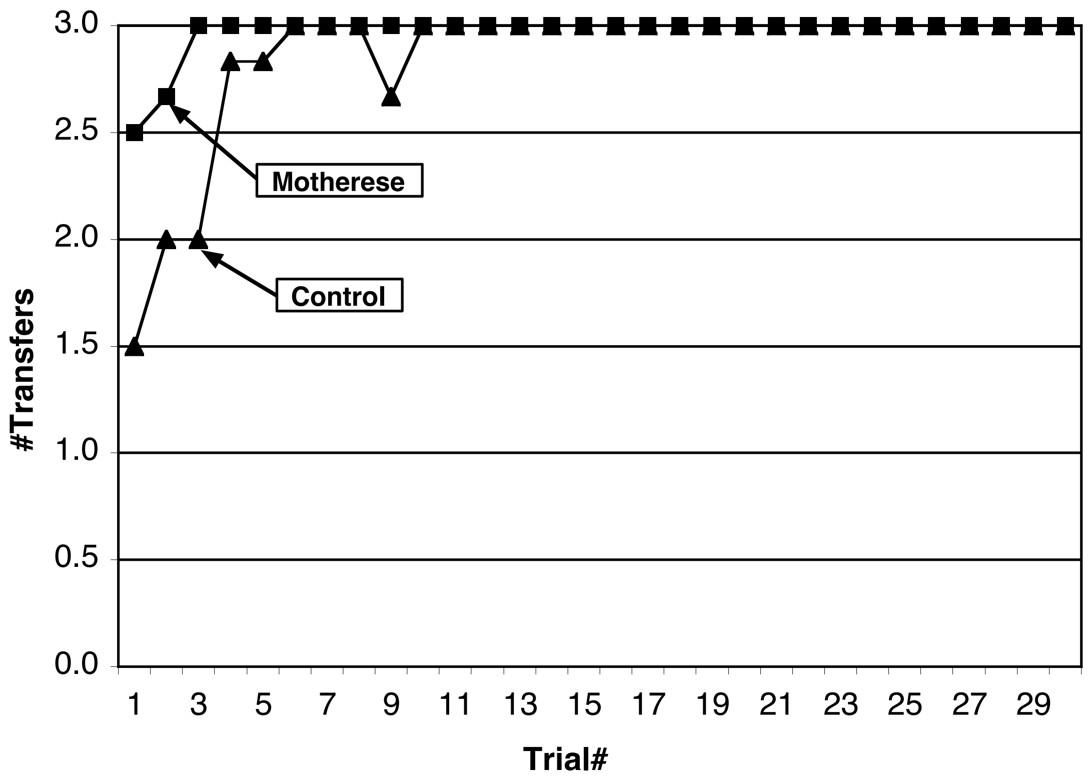


Figure 5.10: This graph shows the average number of transfers completed per trial for each instruction type in block two.

3. The motherese subject group achieves 3 successful transfers for the first time in trial 3.

4. The control subject group achieves 3 successful transfers for the first time in trial 6.

5. The averages for the motherese subject group increase monotonically.

6. The average for the control subject group increases monotonically, with the exception of trial 9, in which it falls from the 3-successful-transfer level for 1 trial.

7. The motherese subject group remains stable at the 3-successful-transfer level from trials 3 through 30.

8. The control subject group remains stable at the 3-successful-transfer level from trials 10 through 30.

Observations 1 and 2 show that although subjects in each instruction group was able to perform transfers successfully in the 1st trial, the motherese subjects completed (33%) more transfers than the control group. By trial 3, each subject in the motherese group was successfully completing all 3 of the transfers in a trial. It was not until trial 6 that a subject in the control group reached this point. Observations 5 and 7 show that the learning period for the motherese subject group lasted for 3 trials, from trial 1 through 3. Although the control group reached the 3 transfer level at trial 6, their average fell from this level for one trial, trial 9, before reaching and staying there in trials 10 through 30. The control group's learning period lasts from trial 1 to trial 10. These observations show that the motherese group's learning period included 7 fewer trials than the control subject group's did. In block two the subjects already had experience with chording and the technique. A positive learning transfer is indicated by two observations, that each group's learning period was shorter in block two than in block one, and that each group's average transfer level was higher in the first trial of block two than in the first trial of block one.

Table 5.1 summarizes the learning period lengths for each of the instruction types in each of the two blocks deduced from the observations made from figures 5.9 and 5.10. In both blocks, the learning period is obviously shorter for the motherese group than it is for the control group. These data support the hypothesis that motherese-aided learning occurs faster than un-aided exploratory learning.

	Motherese	Control
Block 1	8	14
Block 2	3	10

Table 5.1: This table gives a summary of the learning period lengths for the two types of instruction in the two experiment blocks. The values for blocks one and two can be observed in Figure 5.9 and Figure 5.10, respectively.

Variable	Coefficient	Std. Error of Coeff.	t-ratio	Prob
Constant	1310.21	15.76	83.1	HS
Trial #	-1.23538	0.8253	-1.5	NS

Table 5.2: This table gives the results of a regression analysis of *TransferTime* in trials with successful transfers and no mischords. The values of 468 trials were omitted from this analysis, 6 of these values were outliers while the other 422 values were from trials with mischords.

Correctional motherese hypothesis H2:

The transfer technique requires two mouse skills: targeted tracker movement and the clicking of multiple-buttons together, chording. Targeted movement was expected to be well learned by all subjects, especially given the subject group used in the experiment, while chording was novel. Thus it was anticipated that speed improvements in performing transfers would be the result of speed improvements in chording and not in movement. Furthermore, because correctional motherese provides a graphical depiction of chording, it was expected that subjects with motherese would improve more, and faster, in chording and therefore be faster in their transfers.

The first step in evaluating this hypothesis is to ensure that the transfer times do in fact improve, i.e. decrease across the trials in a block. To check this a regression analysis of *TransferTime* by *Trial #* was performed for trials with successful transfers and no mischords, to ensure that no motherese was included in the timings. The results of this regression analysis are given in Table 5.2. Figure 5.11 depicts the trend graphically. The slope of the regression line, the coefficient of *Trial #*, was not significantly different from zero, the null hypothesis. There appears to be improvement, but the regression analysis finds no significant linear change in the

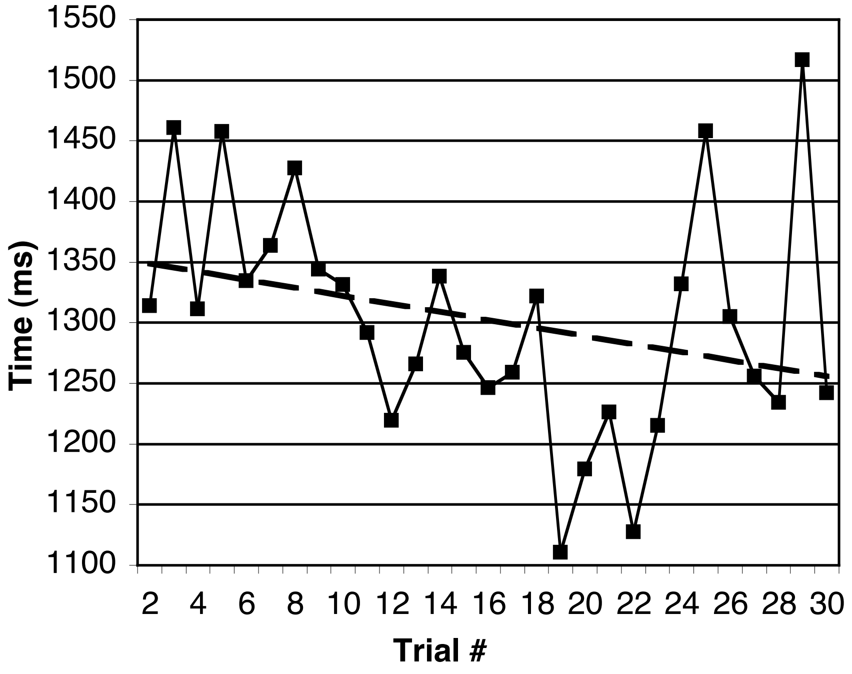


Figure 5.11: This graph shows the average transfer times in the trials with successful transfers and no mischords. Note that there were no such trial 1 trials. The straight, dashed lines show the linear trend, i.e. regression, of these values.

	df	Sums of Squares	Mean Square	F-ratio	Prob
Constant	1	24.1768	24.1768	208	HS
Instruction Type	1	0.834006	0.834006	7.1753	HS
Error	334	38.8219	0.116233		
Total	335	39.65591			

	Average	Cell Count	Std. Error	Prob
Control	0.3202	161	0.0186	HS
Motherese	0.2205	175	0.0186	HS
Difference	0.099729	–	0.0372	HS

Table 5.3: These tables shows the results of a one-way ANOVA of *Instruction Type* on *Mischords* in the post-learning period. 24 outlier trials were omitted from this analysis.

times across the trials, and no significant decrease in the transfer times.

Contrary to expectation, the initial regression analysis found no significant decrease in the transfer times.

Correctional motherese hypothesis H3:

Correctional motherese gives a user corrective feedback when she issues chords that have no effect. It identifies her erroneous behaviour and encourages her to correct it. Its automatic reminders provide an incentive to improve. It gives graphical depictions of the active chords and where they need to be issued, providing users with a visual representation that can be remembered and later recalled when performing the chords again. The combination of the visualization and reminder provided by correctional motherese is expected to cause better internalization of the actions resulting in fewer errors in using them.

The average post-learning mischord rate gives a measure of the accuracy in performing the transfer technique after it has been learned. A one-way ANOVA of mischord rate on instruction type, see Table 5.3, finds a highly significant difference in the accuracy of subjects in the two instruction groups. Subjects with motherese instruction make fewer mischords per transfer in the post-learning period than those without, the reduction being 31%.

The results of the ANOVA support the hypothesis that motherese instruction

	df	Sums of Squares	Mean Square	F-ratio	Prob
Constant	1	22221085.0	22221085.0	1221.6	HS
Instruction Type	1	136304.0	136304.0	7.4936	HS
Error	164	2983080.0	18189.5		
Total	165	3119384.0			

	Average	Cell Count	Std. Error	Prob
Control	402	64	10.75	HS
Motherese	343.2	102	10.75	HS
Difference	58.8734	–	21.51	HS

Table 5.4: These tables shows the results of a one-way ANOVA of *Instruction Type* on *ClickTime* in the post-learning period. 174 values were omitted from this analysis, 4 because they were outliers and 170 because their trials had 1 or more mischords.

improves post-learning accuracy. Motherese instruction resulted in a lower mischord rate than the no-instruction control group. This is the result of better internalization caused by the motherese instruction. The correction and visual representations that correctional motherese provides gives both an incentive and a memory aid to encourage and improve the comprehension of learned techniques. The results show that subjects learn the technique better when they have motherese, but whether this accuracy improvement is due to the visualization or being corrected cannot be determined.

Correctional motherese hypothesis H4:

As discussed in the analysis of H2, the correctional motherese demonstrates the clicking aspect of the technique. The anticipation was that subjects would be clicking faster in performing the technique after initial learning.

A one-way of ANOVA of the average *ClickTime* on instruction type, see Table 5.4, shows that the difference between the post-learning period clicking times for the two instruction types is highly significant. The motherese subject group achieved an average *ClickTime* in the post-learning period that was 59 ms faster than the control subject group's average.

The ANOVA shows that the time spent clicking is significantly, 15%, less for subjects with motherese instruction than for those without. These results support

Variable	Coefficient	Std. Error of Coeff.	t-ratio	Prob
Constant	0.14727	0.1301	1.13	NS
Trial#	0.00237474	0.005528	0.43	NS

Table 5.5: These tables show the results of a regression analysis of the *Mischords* for the motherese instruction group compared to the average. 9 outlier values were omitted from this analysis.

Variable	Coefficient	Std. Error of Coeff.	t-ratio	Prob
Constant	0.366445	0.1451	2.53	S
Trial #	-0.00334059	0.006175	-0.541	NS

Table 5.6: These tables show the results of a regression analysis of the *Mischords* for the control instruction group compared to the average. 26 outlier values were omitted from this analysis.

H6. This effect is expected because the visualization of the active chords and the chording technique, provided by the correctional motherese, helps users to better internalize and perform this novel interaction technique.

Reinforcement motherese hypothesis H5:

Reinforcement motherese is expected to encourage subjects to develop facility after initial learning by replaying (repeating) the technique back to subjects. This was expected progressively reduce the error rate of subjects with motherese instruction compared to those without.

Figure 5.12 shows the average number of mischords made by each instruction type in the post-learning period. Regression lines show the linear trend. The graph shows a relatively large amount of variability in the average mischord rates. The averages are lower for the motherese group on some trials and for the control on others. The control group appears to have more variability as illustrated by its larger range of values and larger trial-to-trial changes. The regression lines show the values converging to a similar value, with the trend of the motherese subject group's mischord rate increasing slightly and the control group's decreasing slightly. The difference between the two lines is quite small. Initially the difference is just over 0.1 mischords and this difference decreases until the end of the trial.

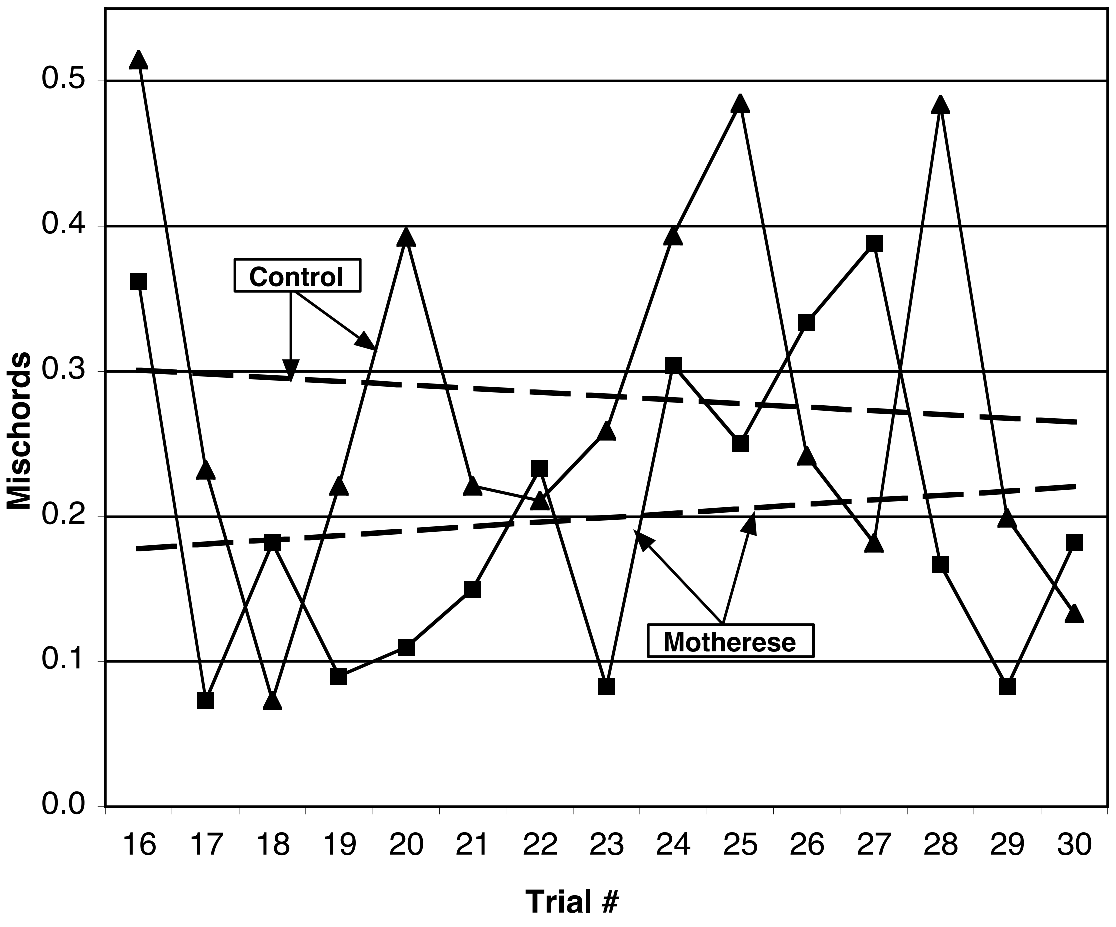


Figure 5.12: This graph shows the average *Mischords* by trial for each type of instruction in the post-learning period trials. The dashed lines show the linear trend of the average mischord rate for each instruction type.

Regression analyses were done to evaluate changes in the mischord rates during the post-learning period. The analyses compute the least squares regression of *Mischords* by *Trial #* for each instruction type. The results of these regression analyses are given in Table 5.5 and Table 5.6. The regression analysis shows no significant linear trend in the mischord values. It is impossible to determine if there is a significant difference between the rates of the two instruction groups.

The results of the analyses for this hypothesis are inconclusive. They are not

Variable	Coefficient	Std. Error of Coeff.	t-ratio	Prob
Constant	1029.19	168.8	6.1	HS
Trial #	9.56375	7.156	1.34	NS

Table 5.7: These tables show the results of a regression analysis of the per-trial post-learning period *TransferTime* for the motherese instruction group compared to the average. 73 values were omitted from this analysis, 2 as outliers and 71 because their trials had 1 or more mischords.

strong enough to reject the null hypothesis. This result does not necessarily imply that there is no difference, but that it cannot be determined from the experiment's results whether one of the instruction types causes faster improvement in the mischord rate or if they both result in the same improvement rate.

Reinforcement motherese hypothesis H6:

Reinforcement motherese was designed to improve learned actions faster than unaided practice. In particular, it plays back the technique in the time users take to complete it to provide a measure of their speed in performing the three phases of the transfer technique. It also highlights the shortest path between the acquired shape and its target to encourage subjects to take the shortest path in the motion part of the technique. By requiring users to watch the entire playback, it gives a penalty for slow performance that is equal to their chording plus movement times. These properties were expected to more quickly reduce the overall transfer time, *TransferTime*, of subjects with motherese.

Figure 5.13 shows the average speed for transfers by trial in the post-learning period made with each type of instruction. The dashed regression lines show the linear trend in the times for each type of instruction. The graph shows a general increasing trend in both conditions, about 120 milliseconds over 15 trials. The slopes of both lines are visually quite similar, suggesting that there is little difference between the changes in transfer times of the groups.

Regression analyses were performed to evaluate linear trends. The results of these analyses for the motherese and control instruction types are given in Table 5.7 and Table 5.8 respectively. The linear regression of the times based on trial number accounts for only a very small amount of the variance in the times (1.7% for the motherese group and 0.9% for the control). The regressions show no significant change in *TransferTime*, and it is impossible to say if the rates of the two instruction

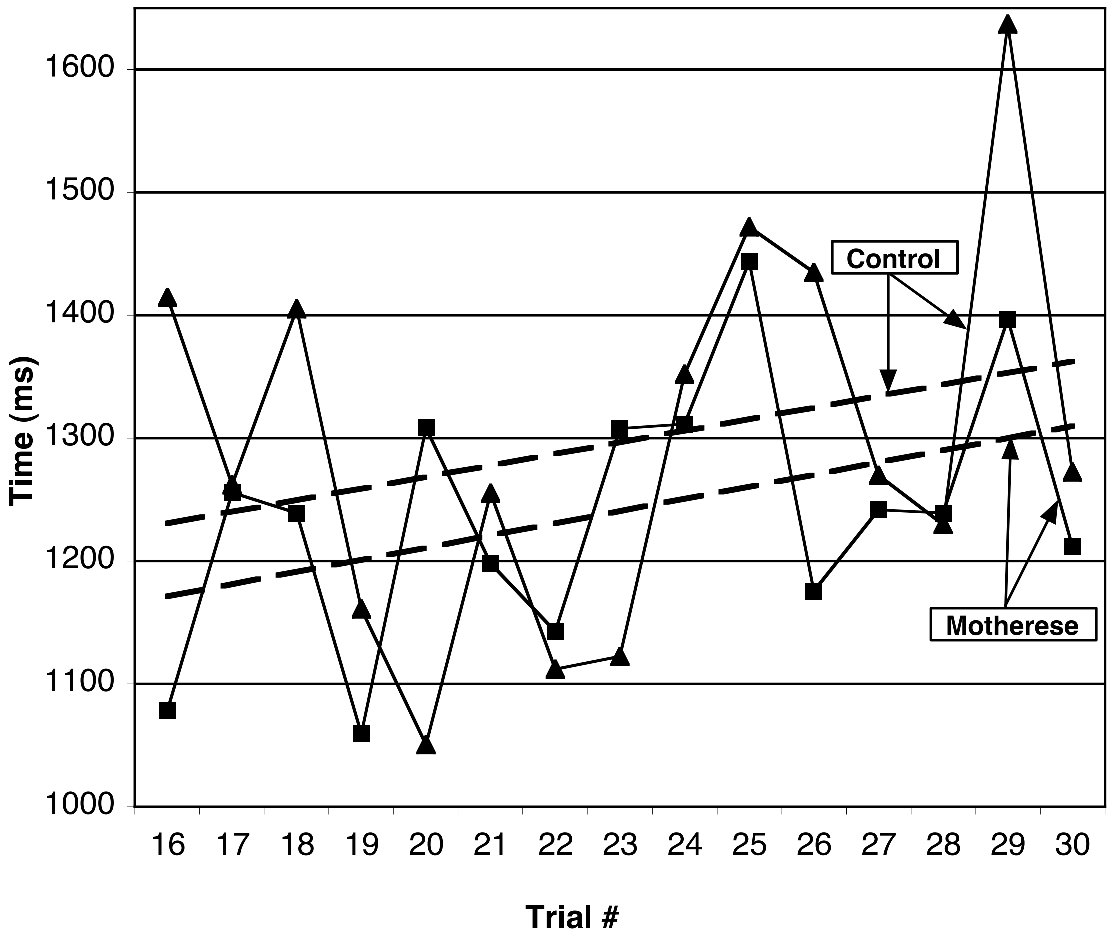


Figure 5.13: This graph shows the average *TransferTime* by trial for the two types of instruction in the post-learning period when no mischords were made. The dashed lines show the linear trend of the times for each type of instruction.

types differ.[4]

Thus, the data are inconclusive also as to if this reinforcement motherese hypothesis holds. It cannot be determined from the data whether the post-learning

[4]The data that could be used for these analyses is significantly reduced by the need to restrict the analysis to trials without any motherese instruction (i.e. those with no mischords) in order to make a fair comparison.

Variable	Coefficient	Std. Error of Coeff.	t-ratio	Prob
Constant	1152.95	211.7	5.45	HS
Trial #	6.71346	8.84	0.759	NS

Table 5.8: These tables show the results of a regression analysis of the per-trial post-learning period *TransferTime* for the control instruction group compared to the average. 115 values were omitted from this analysis because their trials had 1 or more mischords.

period rates of change in the transfer time for the two instruction group's differ. It remains unknown if the reinforcement motherese can result in better post-learning speed improvements in performing the transfer technique.

5.5 Conclusions

The experiment looked at exploratory learning of a chording-based transfer technique supported by motherese-style instructional feedback. Learning performance with this feedback was compared to an un-aided control condition, which provided only traditional interface feedback. Two types of motherese-style feedback were given in the motherese condition. Correctional motherese aimed to help subjects learn the technique faster and internalize it better. Reinforcement motherese was given when the technique was performed correctly to reinforce it and encourage users to speed up their actions.

The hypotheses predicted that correctional motherese would decrease the length of the learning period, increase the accuracy in performing the technique, and increase chording speed. Reinforcement motherese was expected to increase the rate of improvement in the post-learning transfer speed and chording accuracy.

The analysis of the experiment results found that exploratory learning of the technique was greatly improved by the aid of the correctional motherese. The results support the hypotheses that the correctional motherese significantly shortens the length of the learning period. It effectively shortened the learning period both in learning the technique and active chords and in learning just the active chords. It also resulted in better long-term learning of the technique, as demonstrated by significant reductions in the post-learning mischord rate and chording times.

However, the results were inconclusive about the other hypotheses. The analysis was unable to determine if, as anticipated, the reinforcement motherese causes faster improvements in transfer speed and mischord rate, compared to un-aided exploration.

The results also did not find any improvement in the transfer times, as is expected with increased experience. A further examination of this hypothesis remains to be done. The transfer times used in the analysis of H2 contained no mischords, which excluded a large part of the data set. It is hoped that the data for all partially complete trials, removing the portion of their times in which correctional motherese was given, will identify a decreasing trend in the times and allow the determination of how the clicking, motion and transfer times vary relative to one another.

Explanations of these results are provided in the following chapter.

Chapter 6

Discussion and Conclusions

This thesis began by comparing learning how to ride a bicycle with learning how to use an automated telephone system. While both types of learning are user-driven and continue with use, their qualitative aspects are very different. Learning how to ride a bicycle is natural and implicit whereas learning an automated telephone system requires deciphering verbal or written support materials.

The research presented in this thesis looks for ways to make computer user interface learning less like learning the telephone system, and more like learning bicycle riding. It tries to uncover secrets of the self-revealing interface: an interface that implicitly reveals its functionality to users as they use it, with a focus on self-revelation in early interface learning.

Early language learning is proposed as a model for early exploratory user interface learning. The foundation for this model is the strong analogy between languages and user interfaces, and the many similarities observed in a case study. This model supplied the idea to use characteristics of motherese to aid exploratory learning in the user interface.

This chapter discusses the implications of the research presented and directions for future research. Section 6.1 discusses using motherese-style feedback to support user interface learning. Section 6.2 addresses other benefits of using early language learning as a model for early user interface learning. The conclusions of this thesis are given in Section 6.3.

6.1 Motherese

Much of the user interface and artificial intelligence communities believes that computers should be more *human-like*. One result is the creation of online-help assistants, such as the Microsoft Agent (§2.1.1), which use animated characters to make help more approachable. Creating analogies between human language learning and interface learning contributes to this goal. This research supports adding features of human-language learning to interface learning, applying *motherese* to improve user interface learning.

6.1.1 Implications of the experiment

This thesis proposes a technique for increasing interface self-revelation based on motherese. The case study identified a lack of instructive feedback, like motherese, in user interfaces. Interface motherese should aid self-revelation, being user-driven, online, implicit, and integrated within the interface, and also non-verbal, instructing by *showing* instead of *telling*. Providing motherese instruction within the interface frees it from the integration and cognitive load problems that external forms of instruction have as a result of context-switching and translation.

The results of the motherese experiment showed that motherese principles can be used to support interface learning by providing corrective feedback. Motherese instruction successfully shortened the bootstrapping period for a novel interaction technique, and resulted in better post-learning performance.

6.1.2 Limitations of the experiment

The experiment was a preliminary study of motherese-based techniques that support interface learning. It showed that implicit user interface instruction based on motherese can make users learn faster than exploration alone.

The experiment showed certain aspects of motherese that can provide instruction, offering an improvement over un-aided exploratory learning. However, the experiment considered only a single learning situation, a single motherese implementation, a uniform subject group, and utilized only some of motherese's properties. Some questions remain unanswered as to its general applicability. Future studies should evaluate:

- other implementations of interface motherese,

- motherese implementations based on characteristics of motherese that were not evaluated in the experiment, such as *correct response feedback*, which has been shown to be effective for skill retention [39],
- problems implementing effective motherese,
- users' reactions to motherese in response to different types of errors, such as slips, mistakes, and lapses [40].

6.1.3 Future studies of motherese in the interface

This section discusses further studies examining the effectiveness of motherese-inspired feedback in the user interface.

Later stages of learning

Many users do not continue their learning after acquiring the use of an application's basic features. Motherese is a part of the language learning process well beyond learning the basic words and sentence structure, but it changes in response to children's increasing abilities. For example, it changes from simple correction of very young children's incorrect utterances to *extend* their later correct, but simple, ones into more advanced linguistic forms. Perhaps, similarly, motherese-style instruction could encourage users to continue their learning and acquire the use of more advanced actions. If feedback for correct, yet simple, user actions contained information about related, more advanced, functionality, would users be more likely to continue their learning beyond the basics and continue exploring the application's features?

The experiment focused on the effect of correctional motherese, but reinforcement motherese was included also to examine possible improved performance after initial learning. The effects of reinforcement motherese were inconclusive. Possibly, the study was too short for improvements to appear. Possibly the implementation failed to include important features of motherese. Highlighting is a motherese characteristic used to point out important aspects of motherese responses that extend and correct. Highlighting aspects of the interface motherese instruction should similarly help users understand its intentions. Perhaps highlighting the straight path between shapes and the relative speed, for example using sound or colour, would make the experiment's reinforcement motherese more effective.

A long-term study could evaluate the use of motherese in later learning, after the basic functionality of the application has been acquired, to see if user knowledge increases better when motherese is used to subtly suggest more advanced behaviour.

Implementation quality

By giving instruction within the context of an immediate need, motherese provides users with richer instruction than non-context-sensitive learning supports. Hearing children, growing up with deaf parents, lack exposure to spoken language in their immediate environment. Passive observation of language use, via radio or television, does not adequately support language learning, and so children fail to learn [30]. Without the rich communication qualities available in a shared context, children cannot comprehend and model the language they hear. Motherese in the user interface addresses this issue by integrating learning within the interface, the user's working context.

The interface motherese used in the experiment uses abstract animations to demonstrate cause and effect relationships between actions and changes in the application. A recent study [3] shows that watching real demonstrations of others performing motor skills, but without actually performing them, can result in learning. Do these findings apply to motherese? How big an effect does the realness of interface motherese have on learning? Studies could compare the different implementations to real demonstration, to see how well they support learning without practice.

Interruption effects

Extensive testing is always needed when user interruption is included in an interface, because the line between useful and annoying is very thin. Microsoft's original version of the Microsoft assistant (§2.1.1) crossed this line, resulting in a strong negative response and its removal from Microsoft Office.

In the development of the experiment's reinforcement motherese (§5.3.4), small changes were observed to make a big difference in subjects' perception. An early implementation played at a speed that was only loosely based on the user's speed. Users were annoyed and confused being unable to connect their behaviour to what it did. The negative response disappeared once the playback was adjusted to the user's speed. Nearly every subject complained about the original behaviour, but no one complained after the playback time was changed.

In human-human conversation, people expect conversational partners to show an understanding of what they say, and to respond accordingly. Similarly, in human-computer interaction, users expect interfaces to provide feedback about the effects of their inputs. Motherese augments traditional interface feedback with mild correction and suggestions for extension. Motherese in language is natural and not considered annoying [28, p. 279]. Properly implemented, motherese provides guidance that is welcomed, and not annoying. Future studies and implementations of interface motherese can be so designed.

Motherese familiarity

Will the effects of motherese grow as users become accustomed to it and better interpret its intentions, or will they weaken as users learn to ignore it? Performing a longer study in which subjects are exposed to multiple instances of interface motherese can allow this to be evaluated. By counterbalancing the order in which subjects experience the different instances, one can evaluate how its effects are correlated with exposure.

Other help mechanisms

Some might disagree with the choice to leave this consideration to last, arguing that motherese must be compared to existing forms of instruction to show its utility. But, motherese instruction addresses a niche in the spectrum of learning resources: it is user-driven, integrated, purely graphical, demonstrative, and implicit, and therefore differs from the existing forms of instruction. If one insists on comparing its effects to other types of instruction, it should be compared to other online, user-driven resources such as help agents, which are most similar to it.

6.1.4 Interface motherese in practice

As does the work of Carroll and Aaronson [13], this thesis acknowledges that research often lags technology. When this lag occurs, good, empirically validated ideas are not adopted because others have already become standard. User interface self-revelation is not achievable in general, yet cases exist where it is possible and more will be found. This section discusses immediate applications, and what obstructs general interface self-revelation.

Motherese relies on inferring the intentions of conversational partners. While determining user intent in general remains an unsolved problem, some situations exist where it can be inferred, or guessed, accurately enough to be effective. For example, tool-tips use the lingering of the mouse tracker over a component as indication of intent. Spell-checkers provide automatic correction based on a knowledge of common errors and the surrounding context of a word or phrase. The example of the motherese button (§5.2) uses the simple assumptions that clicking on a button widget implies a desire to activate it, and that subjects want to perform actions as efficiently as possible.

The above examples are based only on users' local input behaviour, which is readily accessible, and general assumptions about user behaviour. The difficulty in determining intent arises when its determination depends on knowledge from other sources. Artificial intelligence researchers are trying to solve this problem. Agent research studies the development of autonomous software entities that work in parallel with applications. A sub-area of agent research focuses on agents used to track user behaviour to determine user intentions. The Microsoft Agent [16] is an example. Such agents can decide when to give help, such as motherese. Other artificial intelligence research studies the inference of human emotion through speech and facial expressions, which is facilitated by improvements in the recognition of gestures and speech. This research promises to enable computer applications to determine user intent by enabling the consideration of more qualities of human communication.

6.1.5 Other applications of motherese

This section suggests other applications of motherese.

Self-revealing tool-tips

Standard tool-tips are not self-revealing, as [19] implies. Although they give online, user-guided help by labeling pointed-to items, reading them requires a user's explicit attention, and they can suffer from the terminology problem. Motherese suggests an improvement to address these two issues: *demonstrative tool-tips*. Instead of using text, demonstrative tool-tips show users tool effects using demonstration on the application's current, or a similar, state. This demonstration is removed, like a tool-tip, when the user starts moving the mouse. This idea is similar to animated icons [41] but it has the ability to provide more illustrative depictions of tool effects.

Demonstrative tool-tips avoid the terminology problem and give implicit, detailed instruction through demonstration.

Adaptive help

User interfaces that provide unwanted help frustrate users. Knowing when not to provide help is as important as knowing when to provide it. Online help depends on knowing when and why users make mistakes. Erroneous inputs need to be identified as mistakes, which require instruction, or slips, which do not [40]. Studying the differences in motherese feedback in response to these two classifications of errors can offer ideas about how to detect and handle them in giving interface motherese.

Comfortable learning

Motherese is a *comfort language* [30, p. 129]. It uses soothing sounds, intonation, and melodies to attract and engage the learner [28, p. 279] [30, p. 128–129]. Researchers studying *affective computing* [42] have found that humans experience emotions when interacting with computers. Their research aims to monitor users' emotional responses so as to understand and evoke emotion in users. Studying motherese can help researchers discover mechanisms for invoking emotion that soothes users. One example might be calming users who are experiencing frustration while learning.

6.2 Using the Early Language Learning Model

This research is based on a model of users' exploratory learning, which provides analogies to techniques that enable interface self-revelation. No suitable model existing, this research created one based on early language learning. The language-interface analogy stimulated a search for similar characteristics in language learning. Observations made in the case study of a user's early, unaided exploration supported this hypothesis, supported early language learning as a model for reasoning about early interface learning.

6.2.1 Implications of the model

The early language learning model for early interface learning provides interface researchers with simple, well-studied knowledge about how humans learn through exploration. This knowledge has been accumulated by language researchers over many years. User interface research has, from its beginnings, relied on human-human interaction as a source of ideas. This model formalizes the strong relationship between the two forms of communication and their learning, allowing interface research to use what is known about language learning to reason about and gain ideas for user interfaces learning.

6.2.2 Future directions

Section 4.5 suggests immediate lessons about user interface learning, based on the case study observations. This section offers other ideas to improve interface learning based on the early language learning model.

Learning refinement

Both language learning and user interface learning consist of an initial bootstrapping stage, in which basic usage is learned, and a later refinement stage, in which use is slowly improved and extended. Motherese supports the bootstrapping stage in language learning, and this research has examined its use for this stage of interface learning.

Compared to bootstrap learning, learning in the refinement stage is slower, less intensive, and occurs over a much longer time. Language learning in this stage is supported by both observation and explicit study. Dictionaries, for example, allow learners who already possess a basic understanding to add specific usages to their knowledge. Because explicit study is an important element of the refinement stage of language learning, reference materials need to support query-based learning. In this stage users have acquired basic understanding of interface functionality, which they want to easily augment with specific knowledge.

Using reference materials in language learning is directed towards learning about specific language observed in daily life: while reading, conversing or listening. But most knowledge is gained from such sources does not require reference material, the context in which it is used provides enough information. Thus, the constant exposure to language that most people have allows their language learning to continue throughout life. Experienced interface users also need a source for learning about unknown application functionality after bootstrapping is complete, to continue increasing their knowledge of the interface. But, as mentioned in the previous section, current interfaces do little to support this need. More complex examples of use must be provided to allow learning of interfaces to continue beyond the bootstrapping phase. This could be done by providing action feedback that highlights and demonstrates common follow-up actions, suggesting that users try them in combination with their most recent action.

Context dependence

Language utterances and user interface commands are often ambiguous, disambiguated by context. The disambiguation of spoken language uses rich and redundant information, much of it non-linguistic, that is available to both conversational partners. In human-computer interaction, the context available for disambiguation differs between human and computer, as they differ in perceptual abilities [25]. Applications interpret interface actions based on the application state created by previous inputs. Users interpret feedback and predict the results of their actions based on a model of application state.

However, contextual cues are not always sufficient for disambiguation. Insufficient context or differing understandings of context can result in ambiguities that must be resolved explicitly. When ambiguities occur in spoken language, speakers put their conversations on hold to address them using metalanguage. A conversation is resumed after the intended meaning has been discussed and agreed upon.

Users misinterpret context as a result of being interrupted or ineffective feedback. Interfaces can be designed to prevent ambiguity errors by conveying context effectively and using as few ambiguous actions as possible. Ambiguous actions can be handled by performing default actions while offering other choices, for example using mediators [43]. A pie menu that displays alternate interpretations using a parallel pie [44] is a more implicit, graphical mechanism.

Differing user needs

Characteristically different language is used when conversing with different types of language learners. Motherese is used in conversing with those learning their, to be, native language. Another variant, *foreigner talk*, is observed in conversations with people learning non-native languages. Foreign language learners differ from first language learners as they have a general understanding of language, and only need to acquire the vocabulary and grammar of a new language. The differences between the language learning of children and foreign language learners is similar to the differences in the learning in the two blocks of the experiment (§5.3.2). In the first block subjects had to learn what the technique was and learn the active chords, whereas in the second block they only had to learn the active chords. The existence of differing language variants for interacting with these different language types of language learners suggest that interfaces also interact differently with users when they are learning novel techniques than when learning similar ones. Ideas for adapting interfaces to communicate with various learning needs can be gained from the study of language variants, just as the present research gained ideas from the study of motherese.

Determination of user abilities

Language research has found that the MLU of children is a better indicator of children's language mastery than age (§3.4.4). The model suggests that an analogous measure in the user interface: the *mean length of continuous interaction*, might be an useful measure of a user's skill level. If so, this measure would provide information that could be used to adapt instruction and presentation based on user expertise, eliminating the need for historical user profiles, which are expensive to maintain. For example, a drawing application could measure users' stroke speed, considering slower strokes to be a number of separate movements, and use it to help straighten lines and close strokes for less-skilled users.

6.3 Conclusions

The research presented in this thesis uses the self-revealing process of early language learning to reason about early user interface learning and to devise techniques for making user interfaces self-revealing. It presents a strong analogy between the user interface and language. The case study gives evidence of similarities between early exploratory language learning and interface learning. These observations led to using early language learning as a model for reasoning about user interface learning. The language learning model is a strong, universal example of a self-revealing learning process, from which ideas for self-revelation in the user interface can be drawn. The model was used to identify a need for instructional feedback, like motherese, to support exploratory user interface learning. Motherese provides techniques for supporting self-revealing interface learning through gentle, graphical instruction that corrects and extends user actions. An experiment looked at the use of motherese-based interface feedback and found that, compared to un-aided exploration, it shortens learning time while improving post-learning performance.

This thesis has shown that *early language learning is a good model for studying early user interface learning.*

Appendix A

Analogy Glossary

Spoken language		**User interface**
Shared world (containing both abstract and physical elements)	Domain of discourse	Abstraction of a computer application (presented by its interface)
Two or more people	Agents	A computer application's interface and one or more human users
Facilitate the description and manipulation of a shared world by its human occupants	Purpose	Facilitate the description and manipulation of a program by its user interface and users
Words representing objects, actions, and their properties and relationships	Representations	Graphical elements representing tools, the objects they create and manipulate (i.e. documents), and their properties and relationships
Sequences of words	Combinatorial representations (Utterances)	*User:* Sequences of actions (i.e. invoking tools through affordances or shortcuts) *Interface:* Graphical images composed of graphical elements

Table A.1: This table summarizes the analogy between spoken language and the user interface presented in Chapter 3.

Children's language learning		**Early un-aided exploratory user interface learning**
The utterances of others	Sources for mimicking	The graphical elements of tools: their presentation, context and affordances, and actions known to work in similar situations
Generating observed word usages to gain comprehension of their meanings	Mimicking	Performing observed actions to gain comprehension of their effects
An understanding of words and their use gained from observing the changes in the world and responses from others that they elicit	Comprehension	An understanding of actions and their use gained from observing the changes in the graphical presentations of tools and document states that they elicit
Saying sequences of words that convey intended meanings	Production	Performing sequences of actions that accomplish intended goals
Errors in production occurring when the comprehended meaning of a word or word usage is too general (e.g. all 4-wheel vehicles are cars)	Overgeneralization	Errors in production occurring when the comprehended effects of an action or action usage is too general (e.g. the copy shortcut in all operating systems is Ctrl-C)

Table A.2: This table summarizes the analogy between children's language learning and early un-aided exploratory user interface learning presented in Chapters 3 and 4.

Appendix B

Experiment Information Materials

This appendix contains the information letter, consent form, and debriefing letters used for the experiment.

B.1 Information Letter

June, 2005

Title of Project:	*Exploratory Learning of Mouse-Based Interaction Techniques*
Student Investigator:	Erin Lester
Faculty Investigator:	William B. Cowan University of Waterloo, School of Computer Science (519) 884-4567 Ext. 4548

You are invited to participate in a study that involves the learning of mouse-based interaction techniques. The goal is to learn how to match shapes with their corresponding target shapes using a mouse. As a participant in this study, you will be asked to learn and use this technique to complete a series of tasks. Each task requires you use the technique to fill three targets within a given time period. You will be asked to complete the tasks as quickly as possible. If, on any given trial, you have not completed the task within the given time, the trial will be ended and you will move on to the next trial.

Participation in this study is voluntary, and involves a single session which will take less than one hour of your time. You can stop participating at any time by notifying the researcher. By volunteering for this study, you will learn about the experimental process in general and user interface research. In addition, you will receive a detailed feedback sheet about the purpose of this study. There are no personal benefits to participation (other than possible improved computer skill). You may decide to withdraw from this study at any time by advising the researcher, and may do so without any penalty. All information you provide is considered completely confidential; indeed, your name will not be included or in any other way associated, with the data collected in the study. Furthermore, because the interest of this study is in the average responses of the entire group of participants, you will not be identified individually in any way in any written reports of this research. Data collected during this study will be retained indefinitely, on a secure computer account to which only researchers associated with this study have access. There are no known or anticipated risks associated to participation in this study.

The study will be held at the University of Waterloo in the Computer Graphics Lab, which is located in the Davis Centre, DC 2303.

I would like to assure you that this study has been reviewed and received ethics clearance through the Office of Research Ethics at the University of Waterloo. However, the final decision about participation is yours. If you have any comments or concerns resulting from your participation in this study, please contact Dr. Susan Sykes at this office at (519) 888-4567 Ext. 6005.

Thank you for your assistance in this project.

B.2 Consent Form

I agree to participate in a study being conducted by Erin Lester of the School of Computer Science, University of Waterloo. I have made this decision based on the information I have read in the Information-Consent Letter and have had the opportunity to receive any additional details I wanted about the study. I understand that I may withdraw this consent at any time by telling the researcher without penalty.

I also understand that this project has been reviewed by, and received ethics clearance through, the Office of Research Ethics at the University of Waterloo, and that I may contact this office if I have any concerns or comments resulting from my involvement in the study.

Name (print): ______________________

Signature: ______________________ Date: ______________________

Witness Signature: ______________________

B.3 Debriefing Letter

June, 2005 Experiment Feedback

Dear participant,

I would like to thank you for your participation in this study. The purpose of this study was to examine the effect of a particular style of interface feedback, called "Motherese-style feedback", on exploratory learning — learning in the absence of explicit instruction — in a user interface. You were randomly assigned to 1 of 2 different conditions in which you either did your first trials with or without this particular type of feedback, while learning how to complete the matching task. Motherese-style feedback exhibits the properties of "Motherese", the characteristic and universal way in which adults speak to young children that are learning to speak. It has been shown that children prefer Motherese to regular adult speech. The way in which native speakers talk to people learning their language is also characteristically Motherese. Motherese feedback in language *acknowledges the meaning of incorrect utterances* of language learners while providing correction in the form of the proper way to say what the learner meant. Computer user interfaces are similar to languages in many ways. The hope is that Motherese feedback can be beneficial to users learning in situations where the application knows what it is the user is trying to do.

The data collected during the experiment will contribute to helping to answer whether or not Motherese-style feedback is beneficial, and to further strengthening our argument that language learning is a good model for studying user interface learning.

Please remember that any data pertaining to yourself as an individual participant will be kept confidential.Once all the data are collected and analyzed for this project, I plan on sharing this information with the research community through seminars, conferences, presentations, and journal articles.If you are interested in receiving more information regarding the results of this study, or if you have any questions or concerns, please contact me at either the phone number or email address listed at the bottom of the page. If you would like a summary of the results, please let me know now by providing me with your email address.When the study is completed, I will send it to you. The study should be completed and its results available by the end of July, 2005.

I ask that you please do not discuss this experiment with others until the results have been presented as this may affect their results.

As with all University of Waterloo projects involving human participants, this project was reviewed by, and received ethics clearance through, the Office of Research Ethics at the University of Waterloo.Should you have any comments or concerns resulting from your participation in this study, please contact Dr. Susan Sykes in the Office of Research Ethics at 519-888-4567, Ext., 6005.

Erin Lester

School of Computer Science, University of Waterloo
(519) 888-4567 x7800, eclester@cgl.uwaterloo.ca

Appendix C

Experiment Instructions

Experiment Overview

In this experiment you will complete two sets of trials. The two trial sets differ. Before each set of trials you will be given instructions. After each set of trials a window will appear telling you that the next trial set is about to begin or that the experiment is done. In between each of the individual trials a window will appear letting you know when the next trial will start.

Trials will start automatically, after about 5 seconds.

Between trial sets a window will appear, in which you must click 'OK' to move onto the second set of trials. If you need a break, you can take one at this time. When you are ready to move on to the second set of trials, click the 'OK' button.

Experiment Instructions

In each trial set, your job is to figure out how to use the mouse to pick up copies of shapes and drop them on their (matching) targets.

Targets are UNFILLED rectangles located in the bottom half of the screen. Shapes are FILLED rectangles that match their targets in colour, width, and height, and are located in the top half of the screen.

You will be given a series of trials, each with a set of 3 targets to match. When you successfully pickup a shape, it will have a black border around it. Upon matching a target and shape, the target will become FILLED in to look like the shape. A trial ends when you have successfully matched all its targets or 30 seconds have passed, whichever comes first. There are 30 trials in each set.

[*Motherese condition only:* Watch carefully and play along to learn the technique.]

Complete each trial as quickly as you can.

Bibliography

[1] J. Raskin, *The Humane Interface: New Directions for Designing Interactive Systems*. Reading, Massachusetts: Addison Wesley, 2000.

[2] D. A. Norman, *Design of Everyday Things*. New York City: Basic Books, 2002.

[3] A. A. Mattar and P. L. Gribble, "Motor learning by observing," *Neuron*, vol. 46, pp. 153–160, 2005.

[4] D. Hopkins, "The design and implementation of pie menus: They're fast, easy, and self-revealing," *Dr. Dobb's Journal*, December 1991.

[5] B. Shneiderman and C. Plaisant, *Designing the User Interface*. United States of America: Addison-Wesley, 4th ed., 2005.

[6] J. Rieman, "A field study of exploratory learning strategies," *ACM Transactions on Computer-Human Interaction*, vol. 3, no. 3, pp. 189–218, 1996.

[7] C. D. Wickens and J. G. Hollands, *Engineering Psychology and Human Performance*. Upper Saddle River, New Jersey: Prentice Hall, 2000.

[8] J. M. Carroll, "Minimalist design for active users," in *INTERACT 84 - 1st IFIP International Conference on Human-Computer Interaction* (B. Shackel, ed.), (London, UK), pp. 39–44, September 1984.

[9] R. M. Baecker and W. A. Buxton, *Readings in Human-Computer Interaction*. Morgan Kaufmann, 1987.

[10] J. M. Carroll, R. L. Mack, C. H. Lewis, N. L. Grischkowsky, and S. R. Anderson, "Exploring exploring a word processor," *Human-Computer Interaction*, vol. 1, no. 3, pp. 283–307, 1985.

[11] M. E. Pollack, “Information sought and information provided: an empirical study of user/expert dialogues,” in *CHI ’85: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 155–159, ACM Press, 1985.

[12] M. J. Coombs and J. Alty, “Face-to-face guidance of university computer users - ii. characterizing advisory interactions.,” *International Journal of Man-Machine Studies*, vol. 12, pp. 407–429, 1980.

[13] J. M. Carroll and A. P. Aaronson, “Learning by doing with simulated intelligent help,” *Communications of the ACM*, vol. 31, no. 9, pp. 1064–1079, 1988.

[14] W. C. Hill and J. Miller, “Justified advice: A semi-naturalistic study of advisory strategies,” in *CHI ’88 Proceedings*, pp. 185–90, May 1988.

[15] A. W. Roesler and S. G. McLellan, “What help do users need?: taxonomies for on-line information needs & access methods,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, (Denver, Colorado, United States), pp. 437–441, ACM Press/Addison-Wesley Publishing Co., 1995.

[16] M. Corporation, ed., *Microsoft Agent Software Development Kit.* Microsoft Press, 1999.

[17] N. Ummelen, “Declarative information in software manuals: what’s the use?,” in *Proceedings of the 15th annual international conference on Computer documentation*, (Salt Lake City, Utah, United States), pp. 283–296, ACM Press, 1997.

[18] C. Stieren, “The zen of minimalism: designing a top-of-class manual for beginners and advanced users,” in *Proceedings of the 16th annual international conference on Computer documentation*, (Quebec, Canada), pp. 103–112, ACM Press, 1998.

[19] L. L. Constantine and L. A. D. Lockwood, “Instructive interaction: Making innovative interfaces self-teaching,” *User Experience*, vol. 3, no. 1, 2002.

[20] G. Kurtenbach and W. Buxton, “Issues in combining marking and direct manipulation techniques,” in *UIST91: Proceedings of the 4th annual ACM symposium on User interface software and technology*, pp. 137–144, 1991.

[21] G. P. Kurtenbach, A. J. Sellen, and W. A. S. Buxton, “An empirical evaluation of some articulatory and cognitive aspects of ‘marking menus’,” *Human Computer Interaction*, vol. 8, pp. 1–23, 1993.

[22] J. D. Foley and V. L. Wallace, “The art of natural graphic man-machine conversation,” *SIGGRAPH Comput. Graph.*, vol. 8, no. 3, pp. 87–87, 1974.

[23] R. S. Nickerson, “On conversational interaction with computers,” in *UODIGS ’76: Proceedings of the ACM/SIGGRAPH workshop on User-oriented design of interactive graphics systems*, (New York, NY, USA), pp. 101–113, ACM Press, 1977.

[24] R. Rubinstein and H. Hersh, *The Human Factor: Designing Computer Systems for People.* United States of America: Digital Press, 1984.

[25] R. A. Bolt, “Conversing and computers,” *Human-computer interaction: a multidisciplinary approach*, pp. 694–702, 1987.

[26] J. W. Lindfor, *Children’s Language and Learning.* New Jersey: Prentice-Hall, Inc., 2nd ed., 1987.

[27] J. L. Austin, *How to do things with words.* New York City: University Press, 1962.

[28] S. Pinker, *The Language Instinct.* New York City: William Morrow and Company, Inc., 1994.

[29] R. S. Chapman, “Children’s language learning: An interactionist perspective,” *Journal of Child Psychology and Psychiatry*, vol. 41, no. 1, pp. 33–154, 2000.

[30] A. Gopnik, A. N. Meltzoff, and P. K. Kuhl, *The Scientist in the Crib: Minds, Brains, and How Children Learn.* New York City: Wukkuan Morrow and Company, Inc., 1999.

[31] G. A. Miller, *Spontaneous Apprentices: Children and Language.* New York City: Seabury Press, 1977.

[32] C. Gallaway and B. J. Richards, eds., *Input and Interaction in Language Aquistition.* Cambridge University Press, 1994.

[33] L. Bloom, C. Margulis, E. Tinker, and N. Fujita, “Early conversations and word learning: contributions from child and adult,” *Child Development*, vol. 67, pp. 3154–3175, December 1996.

[34] E. M. Lester, *Personal communication.* 2005.

[35] M. Franzke, “Turning research into practice: characteristics of display-based interaction,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, (Denver, Colorado, United States), pp. 421–428, ACM Press/Addison-Wesley Publishing Co., 1995.

[36] B. J. Pierce, S. R. Parkinson, and N. Sisson, “Effects of semantic similarity, omission probability and number of alternatives in computer menu search,” *International Journal of Man-Machine Studies*, vol. 37, no. 5, pp. 653–677, 1992.

[37] R. Soto, “Learning and performing by exploration: Label quality measured by latent semantic analysis,” in *Proceedings of CHI ’99*, (Pittsburgh, PA), pp. 418–425, 1999.

[38] M. V. Selst and P. Jolicoeur, “A solution to the effect of sample size on outlier elimination,” *The quarterly journal of experimental psychology*, vol. 47, no. 3, pp. 631–650, 1994.

[39] M. M. Sebrechts and M. L. Swartz, “Question asking as a tool for novice computer skill acquisition,” in *CHI ’91: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 293–299, ACM Press, 1991.

[40] J. Reason, *Human Error*. New York City: Cambridge University Press, 1990.

[41] R. Baecker, “Showing instead of telling,” in *SIGDOC ’02: Proceedings of the 20th annual international conference on Computer documentation*, (New York, NY, USA), pp. 10–16, ACM Press, 2002.

[42] R. W. Picard, “Toward computers that recognize and respond to user emotion,” *IBM Syst. J.*, vol. 39, no. 3-4, pp. 705–719, 2000.

[43] E. Saund and E. Lank, “Stylus input and editing without prior selection of mode,” in *Proceedings of the 16th annual ACM Symposium on User Interface Software and Technology*, (Vancouver, Canada), pp. 213–216, November 2003.

[44] M. Terry, E. D. Mynatt, K. Nakakoji, and Y. Yamamoto, “Variation in element and action: supporting simultaneous development of alternative solutions,” in *Proceedings of the 2004 conference on Human factors in computing systems*, pp. 711–718, ACM Press, 2004.

[45] L. Swartz, "Why people hate the paperclip: Labels, appearance, behavior, and social responses to user interface agents," Master's thesis, Stanford University, 2003.