Towards Intelligent, Adaptive Input Devices for Users with Physical Disabilities

Shari Trewin

Ph.D.

University of Edinburgh

1998

Towards Intelligent, Adaptive Input Devices for Users with Physical Disabilities

Shari Trewin

Ph.D.

University of Edinburgh

1998

Abstract

This thesis presents a novel application of user modelling, the domain of interest being the physical abilities of the user of a computer input device. Specifically, it describes a model which identifies aspects of keyboard use with which the user has difficulty.

The model is based on data gathered in an empirical study of keyboard and mouse use by people with and without motor disabilities. In this study, many common input errors due to physical inaccuracies in using keyboards and mice were observed. For the majority of these errors, there exist keyboard or mouse configuration facilities intended to reduce or eliminate them. While such facilities are now integrated into the majority of modern operating systems, there is little published data describing their effect on keyboard or mouse usability. This thesis offers evidence that they can be extremely useful, even essential, but that further research and interface development are required. This thesis presents a user model which focuses on four of the most commonly observed keyboard difficulties. The model also makes recommendations for settings for three keyboard configuration facilities, each of which tackle one of these specific difficulties.

As a user modelling task, this application presents a number of interesting challenges. Different users will have very different configuration requirements, and the requirements of individual users may also change over long or short periods of time. Some users will have cognitive impairments. Users may have very limited time and energy to devote to computer use. In response, this research has investigated the extent to which it is possible to model users without interrupting the task for which they are using a computer in the first place. This approach is appealing because it does not require users to spend time participating in model instantiation. This focus on inference rather than explicit testing or questioning also allows the model to dynamically track an individual user's changing requirements.

This thesis shows that within the context of the keyboard difficulties studied, such an approach is feasible. The implemented model records users' keyboard input unintrusively as they perform their own input tasks. This input is examined for evidence of certain types of input error or indications of difficulties in using the keyboard. In the model presented, conclusions are based on the assumption that the

user is typing English text in a word processing application. However, the design of the model allows any other textual language to be used.

A second empirical study, evaluating the model, is described. The model is shown to be very accurate in identifying users having difficulties in each of the areas tackled, the only exception being those who find a given operation awkward, but are able to perform it accurately. Where it is also possible to evaluate the configuration recommendations made by the model, the chosen settings are effective in reducing input errors and increasing user satisfaction with the keyboard. The model is also able to draw conclusions quickly for users with higher error rates, and shows good overall stability.

In the light of this successful identification of keyboard difficulties, potential applications of the model are suggested. It could be used to help occupational therapists and assistive technologists to assess the keyboard configuration requirements of a new user. It could also be made available to users themselves - many people are currently unaware of facilities they may find useful, and how to activate them. The model could be extended to other areas of keyboard use, and to other input devices. This would allow systems to provide automatic, dynamic support for configuration, which would go some way towards improving the accessibility of computer systems for people with motor disabilities.

Declaration

I declare that this thesis is entirely my own composition, and that it describes my own research.

Shari Trewin

Some of the work described in this thesis was published prior to submission. The pilot study for the investigation into input errors, described in Chapter 4, previously appeared as:

> Trewin, S. (1996) A study of input device manipulation difficulties. *Proceedings of the Second Annual ACM Conference on Assistive Technologies*, 15-22, USA, ACM.

The description of the design of the model in Chapter 7 is partially based on that of the *InputLogger* software, introduced in Chapter 4, and published as:

> Trewin, S. (1998) InputLogger: General-Purpose Logging of Keyboard and Mouse Events on an Apple Macintosh. *Behavior Research Methods, Instruments & Computers*, **30**(2), 327-331.

Further description of the model and its internal evaluation, also presented in Chapter 7, is based on:

> Trewin, S. and Pain, H. (1997) Dynamic Modelling of Keyboard Skills: Supporting Users with Motor Disabilities. In *User Modeling: Proceedings of the 6th International Conference*, A. Jameson, C. Paris and C. Tasso, Eds. Springer Wein, New York, pp 135-146.

Finally, the external evaluation of the model presented in Chapter 9 appeared in condensed form as:

> Trewin, S. and Pain, H. (1998) A model of keyboard configuration requirements. *Proceedings of the Third ACM Conference on Assistive Technologies*, April 15-17, Marina del Rey, USA, pp 173-181, ACM Press. (Winner of the Best Student Paper Award for the conference)

Contents

Figures

Tables

Acknowledgements

First I wish to acknowledge James Skedd, whose determined but frustrated attempts to use a woefully unsuitable keyboard configuration inspired this project.

I am extremely grateful to the University of Edinburgh, who funded the research. The Scottish International Education Trust, AISB Society, Department of Artificial Intelligence, and User Modelling, Inc. also provided funds which allowed me to present papers at the ASSETS '96 and User Modelling '97 conferences.

I wish to thank my principal supervisor, Dr. Helen Pain, for her unfailing enthusiasm, support, feedback and encouragement, and my second supervisor, Dr. Peter Ross, who contributed much in the early stages of the project. I am particularly indebted to my colleague Mike Ramscar, for giving freely of his time in providing thoughtful feedback on drafts of papers and this thesis, regular sanity checks, and challenging and helpful discussions over many a plate of chips! This work has also been influenced by insightful and constructive feedback from many others, particularly Phil Odor, Sally Millar, Paul Nisbet and Allan Wilson at the CALL Centre; researchers at the University of York and the Dundee University Applied Computer Studies Division; and many anonymous conference and journal paper reviewers. I must also thank all my colleagues in the A.I. and Education Group, who have provided an extremely stimulating, supportive and friendly atmosphere in which to work.

My heartfelt gratitude goes to the forty-seven volunteers who took part in my empirical studies, in some cases expending significant physical and mental effort. In relation to the organisation of these empirical studies, the following individuals and organisations were most helpful: the Hands-On-Technology project, Lothian Regional Council, Graeme Glendinning of the Craighall Day Centre, the City of Edinburgh Council Social Work Department, Dr. Annalu Waller, Dr. Nik Whitehead, The Thistle Foundation, and the University's Audio-Visual Services Unit.

I am also grateful to Chris Brew and David McKelvie of the Cognitive Science Department, for providing the digram information used by the model, and to David Lowrie of Computing Services, for invaluable advice on the implementation of InputLogger, and providing the Audit library used in InputLogger.

Finally, thanks to my partner Simon Kelly for knowing when to let me work and when to take me climbing, and providing much practical support.

Trademark Information

No endorsement or recommendation is implied by the mention of any product in this thesis.

Apple is a registered trademark of Apple Computer Inc.

The Audit library is copyright 1992-3 Apple Computer Inc.

ClarisWorks is a registered trademark of Claris Corporation.

CodeWarrior is a trademark of Metrowerks Inc.

Digital is a trademark of Digital Equipment Corporation.

Finder is a trademark of Apple Computer Inc.

IBM is a registered trademark of International Business Machines Corporation.

Macintosh is a registered trademark of Apple Computer Inc.

Metrowerks is a registered trademark of Metrowerks Inc.

Microsoft and MS-DOS are registered trademarks of Microsoft Corporation.

Kinesis is a registered trademark of Kinesis Corporation.

OpenWindow is a trademark of SunSoft.

Power Macintosh is a trademark of Apple Computer Inc.

SimpleText is copyright Apple computer Inc.

SYSTAT is a registered trademark of SYSTAT, Inc.

UNIX is a registered trademark of UNIX System Laboratories Inc.

VMS is a trademark of Digital Equipment Corporation.

Windows and Windows 95 are registered trademarks of Microsoft Corporation.

Windows NT is a trademark of Microsoft Corporation.

XWindow System is a registered trademark of X/Open Company Ltd.

Chapter 1

Introduction

1.1 Motivation - The Domain of Interest

For people with motor disabilities, computer technology has the potential to be a great leveller in the fields of education and employment. It can support both access to information and efficient production of written or graphical work. For this access to be achieved, however, it is necessary for the computer itself to be accessible. As computers become more commonly used in schools, colleges and work places, it is important that their input devices and applications are usable by as many people as possible.

For some users the default input devices - the QWERTY keyboard and mouse - are inappropriate. A host of alternative input mechanisms such as specialised keyboards, switches and voice recognition technology exist (Casali, 1995; Glennen and DeCoste, 1997; Lazzaro, 1995; Nisbet and Poon, 1998). These can be used to circumvent extreme difficulties in using ordinary keyboards and mice.

This thesis focuses on the needs of a different, and possibly larger, class of potential computer users: those who have some motor difficulty which affects their use of the keyboard or mouse, but who nevertheless prefer to use the keyboard and mouse as input devices. The use of these standard input devices has a number of advantages, which will be outlined in Chapter 2.

Several common problems with keyboards and mice are well known. For example, tremor can cause users to press keys or mouse buttons several times when only a single press was intended. One-handed typists may find it awkward to use *modifier*

keys[1] - keys which modify the effect of other keys when they are held down while the other keys are pressed. Difficulty in aiming can make it difficult to point with a mouse or press a specific key on a keyboard.

In order for users with difficulties like these to be able to use input devices with sufficient accuracy and comfort, they must often adapt the way in which they use the devices, or change the way in which the devices behave. Researchers in assistive technology have responded to the adaptation requirements of computer users with motor disabilities by developing many assistive devices and adaptation techniques, both in hardware and software. Of particular interest are *software configuration facilities* - programs built into operating system software which can alter the behaviour of input devices. These facilities will be described fully in Chapter 2. The process of choosing appropriate settings for these facilities is *input device configuration*. Configuration can reduce or eliminate many input errors and other difficulties related to disability (Borden, 1991), although this thesis will show that some classes of error are not well supported by existing facilities.

To date, there has been little formal research into input device difficulties experienced by people with motor disabilities, and the corresponding configuration facilities. In this thesis, much essential groundwork has been laid with respect to further development of the existing keyboard configuration facilities.

Despite the existence of software configuration facilities, several barriers to the successful use of keyboards and mice by people with motor disabilities remain. It can be difficult to find out what facilities are available, how to activate them, and how to choose settings which match the requirements of the user. Configuration itself may require physical control of the input devices beyond the abilities of the user. Currently, little or no automated support for configuration is available.

These barriers between configuration facilities and users constitute the central problem. The work described in this thesis represents an important contribution to a potential solution - automated configuration support.

---

[1] On Page 254 a glossary of technical terms used in this thesis is provided.

1.2 The Approach Taken

Given that human assistance is a limited and precious resource, one way of bridging the gap between users and the available configuration facilities may be to provide some form of automated support for configuration. Configuration support could be built into existing operating systems or applications such as word processors, via their help system.

Adaptive systems, an active research area in both human-computer interaction, and artificial intelligence (McTear, 1993), take the initiative in adapting the interface, or interaction model, to suit the inferred needs of the user, and are of particular relevance to the configuration problem outlined above. Such systems, and intelligent tutoring systems in particular, often base their adaptations on explicit, dynamic *models* of individual users (Benyon and Murray, 1993). This thesis investigates the role that a user model could play in improving the existing support for input device configuration.

Much existing user modelling research has focused on cognitive aspects of users - their knowledge, goals and cognitive skills. For example, McTear (1993) lists a number of types of information to be found in user models, almost all of which are cognitive characteristics. In this thesis, the physical skills and abilities of the user form the domain of interest. Focusing on keyboard use, the thesis examines two aspects of configuration: the choice of appropriate configuration options for a user, and the scope for extension and improvement of the existing configuration facilities.

While user models could be applied in both of these areas, the primary aim of this research has been to develop user modelling techniques for choosing an appropriate keyboard configuration for a user. This thesis presents a model which covers four important areas of keyboard difficulty, and which has been evaluated in an extensive empirical study. The approach taken has been to monitor users' input and draw conclusions from this input, with as little task knowledge as possible. This approach makes no demands on the user, and provides an opportunity to offer configuration support to users who are unaware that configuration is possible.

This thesis shows that such an approach is feasible for several important aspects of keyboard configuration, and the resulting model could form the basis of a very general support system, suitable for use in many different styles of configuration support.

1.3 Aims of the Research

This research had a number of complementary aims, enumerated below in order of priority.

1. To demonstrate the feasibility of using user modelling techniques to identify the input difficulties and configuration requirements of users. This identification was to be achieved without explicit user questioning or testing, in order to produce as general a solution as possible.
2. To gather detailed information about the nature and extent of the difficulties experienced by keyboard and mouse users with motor disabilities, with particular emphasis on input errors. This was necessitated by the lack of existing data, and provided data with which to develop appropriate user modelling techniques.
3. To assess the existing configuration facilities in the light of the information gathered, in order to identify the most important facilities, and gauge their effectiveness in alleviating input difficulties. This allowed the identification of a set of useful and important facilities to be covered by the model.
4. To identify gaps in the existing configuration provision, and if possible to make suggestions for facilities to tackle needs which are not currently addressed. This would provide a foundation from which future research into configuration, and user modelling in configuration could build.

The ultimate aim of this thesis is to contribute to the development and understanding of configuration tools for people with motor disabilities.

1.4 Thesis Outline

The following chapter (Chapter 2) summarises the current state of the art in computer input for people with motor disabilities, with particular emphasis on configuration facilities available for keyboards and mice. It presents the reasons why standard input devices are often preferred over more specialised devices, even by people who have considerable difficulty in using them.

Chapter 3 examines recent adaptive interface and user modelling research, and sketches an outline of how these research areas could be applied to input device configuration.

In the absence of detailed pre-existing data, an empirical study of the use of keyboards and mice by people with motor disabilities was carried out and is presented in Chapter 4. The study focuses on *performance errors* - input errors attributable to physical difficulty in using the devices.

Chapters 5 and 6 relate the observations of Chapter 4 to the configuration facilities described in Chapter 2. Chapter 5 presents new empirical data on the use of two of the most basic configuration facilities.

In Chapter 6 a detailed comparison of the errors observed in Chapter 4 with the available keyboard and, to a lesser extent, mouse configuration facilities is undertaken. This leads to the suggestion of one new configuration facility. A simple prototype of the new facility and some preliminary investigation of its acceptability are described. Chapter 6 goes on to examine the extent to which the available facilities were used by their target population, and the reasons for their lack of use.

Chapter 7 provides a detailed description of the model of keyboard configuration requirements which forms one of the primary contributions of this thesis. This includes an examination of the relevance of existing user modelling techniques to this domain, and the results of internal (formative) evaluation of the model. External (summative) evaluation of the model took the form of a further empirical study, the methodology and results of which are described and discussed in Chapter 8.

Finally, Chapter 9 discusses the research presented, summarising the achievements, identifying limitations, and proposing areas for future research. The potential of this model as the basis for automated configuration support is discussed.

Chapter 2

Computer Input and Motor Disabilities

Today, people with motor disabilities wishing to use computers have, in theory, many options when it comes to choosing an input device. However, despite this plethora of choices, the keyboard and mouse are still preferred by many users. This chapter discusses the input choices open to people with motor disabilities, and the advantages keyboard and mouse use can provide. It goes on to introduce the difficulties keyboards and mice can pose, and ways to overcome, or at least limit the effects of, these difficulties. Existing research on keyboard and mouse use by people with motor disabilities will be summarised.

While mice are a relatively recent invention, the current design of keyboards has evolved over more than two hundred years. Their development has been influenced by pragmatic, mechanical and financial considerations, in addition to ergonomic and usability criteria. Unfortunately, when ergonomics are considered, evaluation tends to involve only non-disabled subjects. As a result, keyboards and mice are not always as easy to use as they could be, and a moderate level of motor disability can seriously affect a user's ability to manipulate them. High input error rates are often the result.

The kinds of input errors tackled in this thesis are often excluded from mainstream research into human error. Throughout this thesis the term *performance errors* will be used to describe errors attributable to physical causes. This term will be defined more precisely and the relation of performance errors to other human error research will be described in this chapter. Finally, this chapter introduces some important practical aspects of the use of these software mechanisms, providing the motivation for the research described in this thesis.

2.1 Keyboards and Keyboarding

Cooper (1983) provides a detailed history of the development of the typewriter and keyboard. He reports that the recorded history of the keyboard can be traced to January 7th, 1714, when Queen Anne granted to Henry Mill a British patent for

> an artificial Machine or Method for the Impressing or Transcribing of Letters Singly or Progressively, one after another, as in Writing, whereby all Writing whatever may be Engrossed in Paper or Parchment so Neat and Exact as not to be distinguished from Print. (Cooper, 1983)

Sadly, no drawings of this machine, or information on its design, have survived.

By the mid-1800's, a number of machines bearing some resemblance to the modern typewriter had been developed. They were large and inefficient, and took longer to produce text than handwriting. Further development was spurred by the clarity and uniformity they offered, and, interestingly, *by their potential for use by people with disabilities* (Cooper, 1983).

The QWERTY layout first appeared in a patented design in 1878. The most commonly cited motivation behind the new layout is the need to eliminate mechanical difficulties with the older, alphabetical layout in which the typebars of common pairs of letters would often jam together when struck in quick succession. The QWERTY layout separated common letter pairs in order to avoid this problem.[1] When technological developments made these mechanical considerations irrelevant, the QWERTY design persisted, primarily due to the large number of QWERTY trained touch typists, and industrial investment in the QWERTY layout.

As mechanical restrictions eased, keyboard development began to focus on improving typing speed and accuracy. In the 1930's, with this in mind, Dvorak developed the major rival to the QWERTY key layout. The design was intended not only to increase typing speed and accuracy, but also to reduce the finger movement necessary for touch typing by placing common letters on the home row. Noyes (1983) and Potosnak (1988) describe a number of influential studies comparing typing speeds on QWERTY and Dvorak layouts, and producing conflicting conclusions. None of these studies have been generally accepted as valid (Greenstein and Arnaut, 1987). The wide exposure of the QWERTY layout makes fair empirical comparison very difficult today.

[1] Alternative theories of the motivation behind QWERTY are summarised in Noyes (1983).

The development of the electric typewriter was another step forward which enabled some people with disabilities to produce writing more easily. For example, in 1952 a man with Parkinson's disease writes:

> For the last twenty years I have been writing, punching the typewriter with my left forefinger - the only digit of the ten that will perform this duty. ... but a snag - inevitably, I suppose - has recently developed. I can no longer type with any ease or dexterity, my one good finger refusing to function. This is really distressing; but, as nearly always happens, there is a way out. In America they are manufacturing an electric typewriter, which at the softest touch does everything, even rolling on the paper and shifting the keyboard; but naturally it is expensive. (Anon., 1952, p55)

As keyboards developed, researchers investigated a number of design features, including key size and shape, keyboard height, size and slope, and the force required to activate keys. Greenstein and Arnaut (1987) and Potosnak (1988) provide summaries of these studies. Typically, researchers have examined the effects of a single variable on input speed or errors, most often for skilled touch typists. To a lesser extent, user preferences have also been examined. Potosnak concludes that "keyboard design has been influenced more by entrepreneurship and convention than by empirical testing" (Potosnak, 1988, p491). She observes that studies on ways in which the different variables interact are particularly lacking. Similarly, Greenstein and Arnaut (1987) report that much keyboard research has been conducted in order to evaluate specific products, and does not isolate the effects of specific features, or combinations of features. Such research is also not usually published in the academic literature.

Potosnak (1988) goes on to observe that "Human performance data indicate that typing behaviour is unaffected across a wide variety of variables, so it is unlikely that further work in this area will uncover factors of significance to the physical aspects of typing" (Potosnak, 1988, p491). While this may be true for experienced typists, it does not necessarily follow that these factors have little effect on typing for other user groups, such as non-touch typists, novices, children and people with motor disabilities. In addition, this conclusion fails to consider potential long-term effects of these variables, including the dangers of typing injuries such as carpal tunnel syndrome.

Concern over such injuries has been increasing steadily in recent years, and research attention has been turned to ways of reducing the risk of such injuries through

improved attention to ergonomics in keyboard design (Hargreaves et al., 1992). In the evaluation of ergonomic keyboards, the force and posture required to operate them, in addition to more traditional measures of speed and accuracy, are important variables (Smutz, Serina and Rempel, 1994). Ergonomic keyboards are not new: the P.C.D. Maltron, for example, was developed in the early 1980s (Noyes, 1983). Such keyboards usually retain the QWERTY layout, but alter the shape of the keyboard and/or the keys themselves. An example is the Microsoft® Natural Keyboard, which has a convex surface, and splits the keys into two sections, one for each hand, in order to reduce wrist flexion for touch typists. The Kinesis® Ergonomic Keyboard also separates the layout into right- and left-handed portions, but has a concave surface for each hand, designed to minimise the digit strength required to reach the keys, and to help the hands maintain a flat, neutral position. A small, short-term evaluation study (Gerard at al., 1994) has suggested that this keyboard is not difficult for trained typists to learn, and that it requires less activity from certain muscle groups than an ordinary keyboard. Gerard et al. conclude that the Kinesis Ergonomic Keyboard may therefore present a reduced risk of stress injuries under heavy use.

Further examples of alternative input devices will be given in Section 2.3.

2.2 Pointing Devices and Pointing

Unlike the keyboard, which has long been the accepted method of entering alphanumeric data into computers, there are a number of popular pointing devices. Each of these has particular advantages and disadvantages, depending on the user, their task, and the environment in which they are using the device.

Some of the more widely used pointing devices are:

- The mouse: A device that the user physically moves across a flat surface in order to produce cursor movement on the screen. The speed at which the cursor moves is a function of the speed at which the mouse itself is moved. This relationship is called the mouse *gain*. On a Macintosh, mice typically have a single button. This may be inset into the upper surface of the mouse, or may be a large area of the upper surface itself. On a PC, the mouse often has two buttons, while on a UNIX® workstation, three buttons are typical. Mouse shapes also vary - the upper and side surfaces may be flat or curved. With a mouse, selection operations are

made by clicking or double clicking, and drag operations are performed by holding down the appropriate button while moving the mouse. One feature of mice is that they sometimes need to be lifted off the surface and replaced in a different position. This occurs when the mouse reaches the edge of the available surface, or the limit of the user's reach, before the cursor has reached the desired position on the screen.

- The trackerball: This device consists of a ball mounted in a base. The cursor is moved by rolling the ball in its casing, and the speed of movement is a function of the speed with which the ball is rolled. Buttons for performing click and double click operations are positioned on the base. For dragging, some trackerballs require a button to be held down while rolling the ball, while others have a specific button which initiates and terminates a drag operation without needing to be held down during positioning.

- The joystick: Joysticks vary widely in design, but all consist of a lever mounted on a base. The lever may be grasped with the whole hand and have integrated buttons, or may be operated with the fingers, with buttons mounted on the base. The cursor is moved by moving the lever in the desired direction. When the lever is released, it returns to its original, central position. On some joysticks (often those with integrated buttons) the speed of cursor movement is directly related to that of movement of the lever, and the cursor returns to the centre of the screen when the lever is released. In these models, the cursor position must be actively maintained at all times, and buttons must be pressed while maintaining the desired position. In other models, the cursor moves at a fixed or steadily accelerating rate in the direction indicated by lever movement, and retains its final position when the lever is released. The buttons are often located on the base of such models, and a drag button is generally included, since it is difficult to hold down a button while moving the lever with a single hand.

- The touch pad: The touch pad is a flat, touch sensitive surface, representing all or part of the screen. The cursor is moved by sliding a finger across the surface in the desired direction. Buttons for object selection are located near the touch surface, and a drag button may or may not be available.

There has been much human-computer interaction research examining the features of these and related devices, and drawing comparisons between them. For example,

Walker, Meyer and Smelcer (1993) examined the trade-off between speed and accuracy when using mice. MacKenzie, Sellen and Buxton (1991) compared a Macintosh® mouse and Kensington trackerball in a series of pointing and dragging tasks. In the Kensington trackerball, dragging is achieved by holding down a button with the thumb while moving the ball with the fingers. It was found that the mouse was both faster and less error prone than the trackerball for both task types. Greenstein and Arnaut (1987) review a number of studies comparing cursor control devices for pointing and tracking tasks, dating from as early as 1967. The findings of these studies have often produced contradictory results for the relative accuracy, speed of use, and even user preferences for these devices. Greenstein and Arnaut conclude that the choice of an ideal device is highly dependent on both the user and their task.

2.3 Alternative Input Devices

For many people with disabilities, the keyboard and mouse are not easy to use. There are literally thousands of alternative devices and software programs designed to help people with disabilities to access and use computers (Casali, 1995; Glennen and DeCoste, 1997; Lazzaro, 1995; Nisbet and Poon, 1998). While a full survey of this field is beyond the scope of this thesis, this section provides a brief overview of the range of alternative input devices available for people with motor disabilities.

Where the keyboard and mouse are inappropriate, a switch system is a popular alternative. A basic switch is a single button which can be mounted in a variety of positions, and operated with whatever part of the body (for example, the head) the user can control well enough. A switch system operates by displaying a menu, or *virtual keyboard*, on the screen. Pressing the switch starts a scan through the menu options, and pressing the switch again selects the menu option currently highlighted. Menu options can include, for example, letters, numbers, whole words, sequences of commands, or sub-menus. It is also possible to control a switch system by sucking and blowing on a device held in the mouth - a "sip-and-puff" device. A number of commercially available scanning systems are described by Lazzaro (1995). Compared with the keyboard, scanning is a very slow input mechanism for text - Brewster, Raty and Kortekangas (1996) report that for some users, each menu item must be highlighted for as much as five seconds. There has been much research on efficient scanning mechanisms, virtual keyboard layouts and other ways of accelerating

scanning input rates (e.g. Brewster, Raty and Kortekangas, 1996; Demasco and McCoy, 1992). Acceleration techniques for switch users include word prediction and macro generation, which will be discussed in Section 2.5.

A switch can also be used to provide input in Morse code. This can be faster than scanning, but requires more accurate control of switch timing, and the ability to remember the codes. Other input mechanisms include eye gaze tracking, head pointing, voice recognition and gesture recognition. A review of these and other input devices in the context of wheelchair and environmental controls is presented by Shaw, Loomis and Crisman (1995). Edwards (1995) reports that the most successful eye gaze systems operate by detecting infra-red light bounced off the user's retina, but there are still many unsolved problems with this technology, such as coping with head movements. Head pointing systems operate by detecting the user's head position and/or orientation using ultrasonic, optical or magnetic signals. Nisbet and Poon (1998) describe a number of existing systems, and note them to be easy to use, providing both speed and accuracy. Gesture recognition, and the recognition of scripts such as proof marks, are active research areas, but this technology has not yet matured to a stage where usable commercial systems can be produced. Baecker, Grudin, Buxton and Greenberg (1995) provide a review of research in this area.

Voice recognition, on the other hand, is more well developed, and is fast gaining popularity as an input technique. In recent years, the price of voice technology has been steadily decreasing, while the quality has increased to the stage where voice input can be a usable and effective input mechanism. Today, several voice recognition systems for the PC are available for less than £100. Lazzaro (1996) reports that "Although speech input and output may not completely replace the keyboard ... it is becoming an interface of choice for many users" (Lazzaro, 1996, p80). There is much active research in human-computer interaction, comparing speech with other forms of input. It is regarded as potentially most useful in task environments where users' hands, and perhaps also their eyes, are occupied. For example, Dillon and Norcio (1997) report an experiment into the adequacy and acceptability of a voice recognition system when used by nurses to record cardiovascular assessment data. Their system was based on a small (70 - 100 word) vocabulary, and was trained to recognise the nurses' voices. They found that user acceptance rates were high, particularly after practice with the interface, and the recognition accuracy was 95 to 100%. However, whether voice recognition offers any advantages over keyboard and mouse input for non-disabled users remains an open question. A review of the often contradictory and

ambiguous studies in this area can be found in Molnar and Kletke (1996). Their own study compared a voice-activated and menu-based interface for a standard software application, and found that users performed less well with and were less accepting of the voice-based version of the interface. Similarly, Damper, Tranchant and Lewis (1996) examined contradictory claims over the superiority of speech recognition in command and control tasks where users are performing more than one task concurrently. They concluded that while the speech based interface was no faster and significantly more error prone than an interface based on abbreviated keyboard commands, the majority of errors were caused by misrecognition of words, and an accurate recogniser would in fact have outperformed the keyboard based interface. Baecker et al. (1995) provide a review of similar voice recognition studies.

Accuracy is not the only potential barrier to uptake of current speech recognition systems in general purpose computing. The requirement to dictate aloud, especially in a shared space, may also be off-putting to potential users. While manufacturers claim that input speeds of 70 to 100 words per minute can be achieved, initial results are often disappointing, and some users may give up before the system has adapted to their voice (Nisbet and Poon, 1998). For users with speech impairments, error rates may be unacceptably high. Nisbet and Poon also note that some users have reported voice strain from using this input technique. Speech input is a very useful alternative for some users who find keyboards and mice difficult to use, but is not likely to become the default computer input mechanism at least in the immediate future. While future improvements in speech recognition technology may lead to more mainstream acceptance of voice as an input mechanism, it appears that this is some way off.

For some people, the standard QWERTY keyboard is difficult to use, but an alternative keyboard might be adequate. Many different alternative keyboards are available, including:

- ergonomic keyboards shaped to reduce the chances of injury, and to increase comfort, productivity and accuracy;
- over-sized keyboards with large keys which are easier to isolate;
- under-sized keyboards which require a smaller range of movement;

- one-handed keyboards shaped for left- or right-handed operation. These may have a full set of keys, or a reduced set which are pressed in combinations in the same way as a woodwind instrument is played;

- membrane keyboards which replace traditional keys with flat touch-sensitive areas.

Lazzaro (1996) describes a number of alternative keyboards for the PC.

Users who have difficulty in controlling the mouse also have many alternative ways in which they can control the cursor. The trackerball, joystick and touch pad have been described in Section 2.2. The eye gaze technology mentioned previously is also appropriate for cursor control. Keyboard cursor control is also possible, and will be described in Section 2.6

2.4 Trade-offs: Choosing an appropriate input mechanism

Despite the many different keyboards and alternative devices available, in practice there may not be a solution which meets all of a particular user's requirements. Factors limiting the available choices can include:

- Physical considerations: Ideally the chosen device should not demand any movements which the user finds difficult or tiring. It should allow a reasonable level of input accuracy. Speech recognition, for example, can be error prone.

- Task considerations: The device should allow full access to the functionality of the desired applications. Unfortunately, "alternative input devices alone, even the most sophisticated, are not sufficient to allow disabled people to use existing mass market applications; very few are adapted for anything but mouse or keyboard input" (Shaw, Loomis and Crisman, 1995, p263). For example, Anderson and Smith (1996) observe a number of difficulties with the interfaces of mainstream music composition systems, as experienced by users of non-standard input devices. It has been predicted that in the future, input mechanisms such as voice recognition, gesture recognition and eye gaze technology, will become increasingly mainstream (Jacob, 1996; Zeigler, 1996). Whether they will do so in a form which allows their use by people with motor disabilities remains to be seen.

- Temporal considerations: Input should be as fast as possible, without sacrificing accuracy. For example, switch interfaces can be frustratingly slow to use.

- Financial considerations: The chosen device must be within the budget of the individual, or the organisation providing the computing facilities. For example, head controlled pointing systems are currently priced in the range £995 to £1635, while a trackerball with buttons for drag and double click can cost between £40 and £265 (Nisbet and Poon, 1998; SEMERC, 1998). Unfortunately, many organisations providing computing facilities specifically for people with disabilities have very limited funding. In a recent interview for the British Computer Society Disability Group Magazine (Aeberhard, 1998), employment minister Alan Howarth acknowledges that there is currently a gap in government funding of computer equipment for people with disabilities outside formal education or employment. One computer centre manager at an Edinburgh day care centre reported that the day centre was in urgent need of basic facilities such as lifting equipment in the bathrooms, and that new computer software or equipment was a luxury they could not afford (Graeme Glendinning, Computer Room Manager, Craighall Day Centre, personal communication).

- Environmental considerations: The device should be appropriate for the environment in which it is to be used. For example, a voice recognition interface may not be appropriate in a noisy environment.

- Portability: People wishing to use computers at home and at a workplace or college would find a portable device more useful, and would prefer to be able to use similar set-ups in different locations, rather than being tied to a single non-portable machine.

- Personal considerations: Users may have strong personal preferences. In the words of Edwards (1995): "People who are labelled as disabled have already been singled out. They do not want to be marked as different from their peers any more than is necessary. As far as is practical, they want to use the same equipment" (Edwards, 1995, p26). An example of this is Vanderheiden's (1985) account of the use of special typewriter arrangements by one-handed typists. These key arrangements were designed to reduce the movement necessary to operate the typewriter with one hand, but "the use of rearranged typewriters made some individuals feel that they were treated as inferiors by their peers, and that their use

> overemphasised the negative aspect of their disability ... Those who used standard typewriters seem to feel that they were actually treated with greater respect because they not only did the same type of work on the same equipment as their peers but did so with a handicap as well" (Vanderheiden, 1985, p272).

Unsurprisingly, these factors often conflict, and the assignment of priorities between them can only be done by the individual who is choosing a device. The keyboard and mouse are an attractive choice in terms of access to application functionality, input efficiency and cost. They can be used in many different environments, and are available, in more or less the same form, on all different computer platforms - skills learned at home can be transferred to machines at work, college, or other public places. Finally, their use does not identify the user as different, or disabled. It is not surprising, then, that many users who are on the borderline in terms of the physical ease with which they can use keyboards and mice still continue to use them. It is this class of users - those who wish to use standard input devices, but experience some difficulty in doing so - with which this thesis is concerned. The following section outlines some of the facilities available to support keyboard and mouse users with motor disabilities.

2.5 Improving Keyboard and Mouse Access

Many less severely impaired users can use standard input devices with minor modifications. A number of simple mechanical aids are available which can enhance the usability of ordinary keyboards and mice without changing the input devices.

For some people, positioning is very important. Adjustable tables allow keyboard and screen height to be adjusted, and this in itself can have a dramatic effect on input accuracy. The keyboard tilt can also be adjusted. For those who find it tiring to hold their arms above the keyboard or mouse, arm supports can be fitted to tables. Wrist rests can also provide a steadying surface for keyboard use. Some users wear finger or hand splints while others use a prodder or headstick to activate keys. For those who accidentally press keys, and those who wish to rest their hands on a steady surface, a keyguard can be useful. This is a clear plastic sheet which fits over the top of the keyboard. It has a hole for each key, and keys are activated by pressing through the appropriate hole. Some users find that keyguards are very comfortable and improve both speed and accuracy of their typing. Others find that they slow down their typing, and they can make it difficult to see the letters on the keys. For one-

handed typists, mechanical latches can be used to hold down modifier keys while other keys are pressed. It is also possible to purchase mechanical devices which attach to a mouse and damp out tremor.

The potential effectiveness of physical modifications to input devices is illustrated by Treviranus et al. (1990), who describe three case studies of users for whom modification of standard pointing devices was required. They define the physical demands made by direct manipulation interfaces, and the difficulties these caused for three users with disabilities. In all cases the final solution involved a combination of pointing devices, or minor modifications to a standard device.

With the exception of adjustable tables, modifications like these are usually cheaper and more portable than specialised input devices. Most do not commit the user to a particular machine, and are reasonably discreet. Even more subtle, however, are software facilities designed to alleviate difficulties and reduce errors. Software programs can be used to alter the behaviour of input devices, or the input requirements of applications. Software modifications can tackle input errors by changing an input device's response to specific inputs. They can reduce fatigue by reducing the volume of input required, or they can provide alternatives to difficult movements. They can also minimise the input required of the user, thus reducing effort and opportunities for error. Some examples of useful facilities that can be implemented in software are:

- Keyboard and mouse configuration: the way the keyboard or mouse reacts to a given input can be changed. For example, the key repeat delay can be altered, or the cursor can be made to move more slowly relative to the mouse. In this thesis, such facilities are referred to as *configuration facilities*, or *software configuration facilities*, and the process of finding and choosing appropriate settings for these facilities is described as *input device configuration*. Some of the options available are software versions of existing physical accessories such as key latches, others are novel. Simple alterations like these can be very effective, and are the main focus of this thesis. The following section describes the available configuration facilities in more detail.

- Macro generation: for users who find input slow or laborious, macros can be used to perform common sequences of operations with a single command. For example, a user who always entered the same application and opened the same file after logging on to a system could define a macro to open the file automatically.

Many word processing packages also include macro facilities, to allow commonly used text to be reproduced quickly. For example, a user could create a macro representing their address as it appears at the top of a letter.

- Word prediction: this is a technique intended to reduce the number of keystrokes required when typing text. Many word prediction systems have been developed, and a number are reviewed in Millar and Nisbet (1993). As a user begins to type a word, the prediction system offers suggestions for what the word might be. If the desired word is suggested, the user can choose it with a single command. In practice, word prediction systems have been observed to reduce the number of keystrokes required by up to 60% (Newell et al., 1995). Newell et al. (1995) report that using the PAL word prediction system, some users were able to double their input speed. However, studies with disabled users have also shown that a reduction in keystrokes does not necessarily produce an increase in input rate (for more detailed summaries, see Horstmann and Levine, 1991; Horstmann Koester and Levine, 1994). Word prediction is most useful for very slow typists, particularly switch users. Those who type at a rate of greater than around fifteen words a minute may find that the time spent searching the lists of suggestions for the right word is greater than the time saved (Millar and Nisbet, 1993). Nevertheless, faster users may still find word prediction helpful in reducing fatigue, reducing errors, or improving spelling (Millar and Nisbet, 1993).

Input acceleration techniques such as word prediction and macros are in many ways a complementary approach to configuration of the input devices themselves. While they can reduce errors, they do require a certain level of accuracy on the part of the user in order to be effective. Configuration, on the other hand, is primarily a mechanism for improving accuracy through reducing errors, rather than increasing input rate. For users with slow input rates, or those who tire easily, both techniques can be useful.

2.6 Operating System Software Supporting Physical Access

As mentioned in the previous section, there are a number of simple adjustments that can be made to the way in which the keyboard and mouse respond to input, which can have a profound effect on the accessibility of these devices. These adjustments can be

implemented in software, which has a number of potential advantages over using specialised input hardware. One such advantage is portability between machines running the same operating system: software can be transferred easily on disk or over the internet, giving users access to many machines without having to carry heavy equipment which may get damaged in transit. Software facilities are also discreet: to a casual observer, the system looks the same as any other. Finally, software of this kind is often free, or relatively cheap in comparison with specialised hardware.

Historically, software configuration facilities first appeared as third party software patches which users could install on their systems in order to modify the behaviour of the keyboard or mouse (Novak et al., 1991). An example is the 1-Finger program, released by the Trace R&D Center at the University of Wisconsin Madison in the early 1980s. It was an MS-DOS® application which allowed the use of modifier keys with separate, rather than overlapping key presses, and also allowed users to control the repeating behaviour of keys (Novak et al., 1991).

Novak et al. (1991) note that the effectiveness of such programs was often dependent on the mechanism for retrieving keystroke information used by the active application. Conflicts with other software patches were possible, and the software required upgrading for each new hardware platform and new version of DOS. Many of these difficulties could be solved if the facilities were implemented as part of the operating system itself. Manufacturers would then assume responsibility for upgrading the facilities in line with the operating system, and the facilities could be implemented at the most appropriate level in system software. This solution also helps to ensure that information on accessibility is provided with machines as they are shipped, and included in on-line help systems, making it easier for new users to find out about the facilities. The Trace R&D Center and others have been active in pursuing this goal, and their efforts in lobbying operating system manufacturers have been very successful. Today, many access facilities have been embedded into operating system software.

Software configuration facilities are available in one form or another for the majority of popular operating systems. The remainder of this section summarises the keyboard and mouse configuration facilities available in DOS, the Macintosh operating system, Windows® 3.1, Windows NT™, Windows 95®, and UNIX operating systems.

2.6.1 DOS

One of the earliest systems for which access facilities were available was IBM® DOS.

IBM provided support to the Trace R&D Center, and they developed AccessDOS (Borden, 1991), a collection of eight utility programs intended to improve accessibility for people with mobility, vision and hearing disabilities. AccessDOS has been distributed free of charge by IBM since May/June 1991 and is available for DOS 3.3 and later versions.

Once installed, AccessDOS is a menu-based system operated by using the cursor keys, or keyboard shortcuts for menus and menu items. It contains five utilities relevant to keyboard and mouse users with motor disabilities:

- ***StickyKeys*** - is a facility useful for people who find it difficult to press two keys at once. This is generally required when using the modifier keys: *Shift*, *Control* and *Alternate* on IBM PCs. For example, to produce an asterisk, it is normally necessary to hold down the *Shift* key while pressing '8'. With *StickyKeys*, modifier keys can be pressed separately, so an asterisk would be typed by pressing and releasing *Shift*, and then pressing '8'. This is particularly useful for people who type with one hand, one finger or with a stick attached to their head or hand.

  Sticky Keys can be activated by using the menus of the AccessDOS utility, or simply by pressing the *Shift* key five times in a row. A long beep, rising in pitch, confirms that the utility is activated. The utility can be deactivated in the same way, and a long beep falling in pitch confirms deactivation.

  When activated, modifier keys can be latched or locked. When a key is latched, it affects only the next alphanumeric key press. A modifier key is latched by pressing it once. Because the latch remains active until a key that is not a modifier key is pressed, keystroke combinations requiring two or more modifier keys can be performed easily by simply pressing the required modifier keys in turn, and then pressing the key to be modified.

  When a modifier key is locked, it affects all key presses until it is unlocked. Modifier keys are locked by pressing them twice and unlocked by pressing them once.

By default, AccessDOS uses sound to inform users of events such as keys being latched, locked and modified. These sounds can be disabled using the menus in the utility itself. These menus also allow configuration of other aspects of the utility's behaviour. The locking facility can be disabled. The utility can also be set to deactivate whenever a modifier key is pressed simultaneously with a non-modifier key. This facility is intended to be useful on shared machines where only some users may need the facility. Other users need never know it exists as it will be deactivated when they start to use modifier keys in the default way.

- ***MouseKeys*** - is a useful utility for people who can use a keyboard, but not a mouse. It allows users to control the on-screen pointer using the keyboard, including click, double click and drag operations. This is achieved by using the numeric keypad on the keyboard.

  MouseKeys can be activated by using the AccessDOS menus, or by pressing *Left Alternate*, *Left Shift* and *Num Lock* at the same time (*Sticky Keys* can be used to do this where simultaneous key presses are difficult). The utility can be deactivated in the same way. As with *StickyKeys*, long rising and falling pitches confirm activation and deactivation. When activated, the *Num Lock* key will toggle the numeric keypad between acting as a numeric pad and a mouse pad.

  MouseKeys assumes a standard IBM extended keyboard layout and an IBM PS/2 or Microsoft mouse. Other keyboards with a numeric keypad can be used, but there can be difficulties if no mouse, or a different mouse, is present. When an appropriate mouse is present, the mouse and numeric keypad can both be used, but not in all applications, and not if the mouse is attached to the serial port. These caveats illustrate the difficulty third-party software developers encounter when trying to provide low-level software extensions to operating systems.

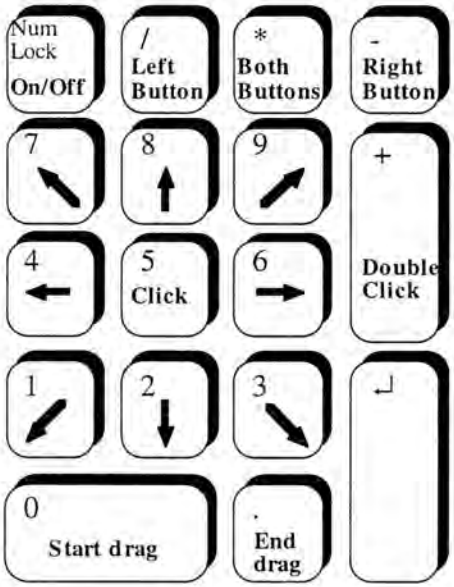


Figure 2.1: Using the numeric keypad to control the pointer with *Mouse Keys*

The mapping of keys to mouse operations when using *MouseKeys* is illustrated in Figure 2.1. The numbers 1-4 and 6-9 move the mouse in eight different directions, the direction being determined by the relative position of the numbers around '5', the central key on the keypad. The pointer begins moving slowly, and then accelerates to a maximum speed. The number '5' itself performs a mouse click. However, the PS/2 and Microsoft mouse both have two buttons. The default button is the left button. Users can also specify the button to be used, before performing a mouse click. This is achieved by pressing either '/' to activate the left button or '-' to activate the right button. On 101-key keyboards, pressing '*' makes both buttons active.

Double clicks can be performed on either button by pressing the '5' key twice in quick succession, or by pressing the '+' key once. The '0' key locks the chosen

button down until it is released by pressing the '.' key. This allows drag operations to be performed.

Using the menus in the AccessDOS program itself, some further features of the utility can be controlled. The sounds can be enabled or disabled, the maximum speed at which the pointer moves can be controlled, and the rate at which the cursor speeds up when moving can be set within the range one to four seconds.

- ***RepeatKeys*** - allows users to control the auto-repeat feature of the keyboard. This is the feature which causes characters to be repeated when the corresponding key is held down for long enough. This feature can cause difficulties for users who cannot press keys quickly, since unwanted repeated characters frequently appear. The following real example shows the characters generated in typing the word "with", for someone whose key press length tends to produce two copies of each character. Individual keystrokes are separated by spaces:

  ww <delete><delete> ww <delete> ii <delete><delete> i tt hh <delete><delete> <delete><delete> t hh <delete><delete> hh <delete>

  The word was eventually correctly typed, using sixteen keystrokes. Had this person opted to use a spelling checker instead of making manual corrections, the word presented to the checker would have been "wwiitthh" The spelling checker for Microsoft Word 6 makes the suggestions "wittier" and "witty", given this input - production of the correct word would have required manual editing of these suggestions, reintroducing the original problem. A better solution is to eliminate such errors at source. *RepeatKeys* solves the problem by allowing users some control over the length of the delay before keys begin to repeat, and the rate at which they repeat. Repeats can also be turned off altogether.

  RepeatKeys is one of three keyboard response facilities. The desired *RepeatKeys* settings must be chosen by using the menus in the AccessDOS program. These settings, together with the settings chosen for the other two facilities in the group, are activated and deactivated by holding down the *Right Shift* key for eight seconds.

- ***SlowKeys*** - is the second keyboard response facility. It is intended to allow users to eliminate accidental short key presses made, for example, by bumping other keys while moving to a specific key. Usually on a keyboard, a character is registered as soon as a key is pressed, regardless of how long it is pressed for. *SlowKeys* allows the user to introduce a key acceptance delay, so that only key presses longer than the delay will cause characters to register.

  Using the AccessDOS menus, the user can control the delay imposed, or choose "no delay" (equivalent to turning *SlowKeys* off). The user can also choose whether or not to hear a click every time a key is accepted. Along with the chosen *RepeatKeys* setting, the *SlowKeys* setting is activated and deactivated by holding down the *Right Shift* key for eight seconds.

- ***BounceKeys*** - is the third keyboard response facility. It is intended to help people who have a tendency to 'bounce' on keys: that is to press them two or more times in quick succession, when they only intended to press them once. Users with shaky hands are particularly likely to bounce on keys. *BounceKeys* allows the user to introduce a delay after each key press (the *debounce* time), during which time the same key, if pressed down, will not register, while a different key will register immediately. This means that the unwanted extra copies of letters will not register. If the user wishes to type the same letter twice, however, they must be sure to leave a long enough gap between typing the letters, so that the second copy of the letter will register.

  BounceKeys cannot be used with *SlowKeys*, and AccessDOS will issue a warning if both are set. *BounceKeys* settings must be chosen using the menus in the AccessDOS program, and the settings are activated and deactivated in the same way as those for *RepeatKeys* and *SlowKeys*.

Some users may have difficulty in accessing the AccessDOS menus using the default keyboard settings. For example, someone who presses keys for a long time would have difficulty in making choices from the menus, since this requires accurate positioning of the cursor by using the arrow keys. In order to allow such users to choose appropriate settings with the AccessDOS menus, an "emergency enable" facility is provided. This is activated by holding down the *Right Shift* key to activate the keyboard response group, and then keeping it held down once the group activates after eight seconds. After a further four seconds, the auto-repeat facility is disabled

and the *BounceKeys* setting is set to one second. After four more seconds, *BounceKeys* is disabled and *SlowKeys* is enabled with a maximum acceptance delay of two seconds. With one of these two options, while the keyboard may be slow and difficult to use, it should at least be possible for most users to access the AccessDOS utility and choose more appropriate settings.

AccessDOS allows users to save settings between sessions, and to implement a time-out period, after which AccessDOS will be turned off. This is useful for shared machines. It comes with comprehensive, clear instructions (Borden, 1991) on using the utilities, and on tackling difficulties that may be encountered when installing AccessDOS on different platforms.

2.6.2 The Macintosh Operating System

Apple® was in fact the first major vendor to incorporate access facilities into their operating system. Since 1987, the "Easy Access" control panel has been available for all Macintosh machines. Although it is currently not part of the default operating system configuration, it is included on the system disk shipped with new Macintosh machines. While the on-line documentation system is not ideal for novice users, and requires good control of the mouse, it does include brief but clear instructions on the use of the facilities. These can be found by searching for the keyword "keyboard" and choosing "How do I adjust the way the keyboard works?".

Easy Access currently includes:

- *Mouse Keys*: This allows the cursor to be controlled by the keyboard, and is very similar to the AccessDOS *MouseKeys* utility. Because Macintosh mice have only one button, there are no commands for choosing a left or right mouse button. As in AccessDOS, movement in eight directions, a click button, and drag buttons are provided. There is no double click button. Users can adjust the delay before the cursor starts to move (five choices, "long" to "short"), and the maximum speed at which it moves (eight choices, "slow" to "fast"). When Easy Access is installed, users can activate *Mouse Keys* with the keyboard shortcut '*Apple-Shift-Clear*'.

- *Slow Keys*: Performs the same function as the AccessDOS *SlowKeys* utility, providing five choices ("long" to "short") of key acceptance delay. It can be

activated and deactivated by holding down the *Return* key for ten seconds. Users have the option of hearing a click when a key is accepted.

- *Sticky Keys*: The Macintosh version is almost identical to the AccessDOS version, providing locking and latching of modifier keys, and using five *Shift* key presses as a keyboard shortcut. Users have the option of hearing a click when a modifier key is pressed. On a Macintosh, there is a fourth modifier key affected by *Sticky Keys*: the *Apple* key. There is one important difference between the Macintosh and AccessDOS versions of *Sticky Keys* - on a Macintosh, pressing a modifier key simultaneously with an alphanumeric key will automatically deactivate *Sticky Keys*. This behaviour cannot be suppressed, and is not described in the Macintosh on-line help. It is also not possible to suppress the locking mechanism, which can be done in AccessDOS. Both of these features can cause difficulties for users. This point will be revisited in Chapter 5.

The Macintosh operating system also allows users to control the key repeat delay (five choices, "off" and "long" to "short") and key repeat rate (five choices, "slow" to "fast"), as *RepeatKeys* does in AccessDOS. These options are part of the "Keyboard" control panel, which is included in the default system installation.

In addition, the "Mouse" control panel, also a part of the default installation, allows users to control the mouse tracking gain. Seven choices - "very slow" and "slow" to "fast" are available. Users can also adjust the double click speed between three values. This is the maximum time allowed between the two clicks of a double click. Searching the on-line help for the keyword "mouse" and choosing "How do I adjust the mouse or the trackball?" provides information on these settings. For potential *Mouse Keys* users, it is unfortunate that the choice "Why can't I move the pointer on the screen?" provides no information on *Mouse Keys* or alternative pointing devices.

The functionality of *Bounce Keys* is not currently available for Macintosh machines.

2.6.3 Windows 2.0/3.0/3.1, Windows NT

Lee (1989) describes how the implementation of PC-DOS and MS-DOS access software such as AccessDOS relied on the ability to run "Terminate and Stay Resident" (TSR) programs, and to hook into device drivers and software and hardware interrupts within the operating system. The majority of these programs do not work with

Microsoft Windows, since it takes control of many interrupts and does not leave TSR programs operating. When Microsoft Windows began to gain popularity on PC-DOS and MS-DOS platforms, new versions of the access utilities were required. Lee (1989) describes the Trace R&D Center's implementation of AccessDOS functionality for Windows 2.0, and the many technical challenges this posed. Microsoft Corporation supported this work, and the result was "Access Pack for Microsoft Windows", first released in October 1990, and currently available for Windows 3.0, 3.1, 3.x and Windows NT.

2.6.4 Microsoft Windows 95

Windows 95 has all of the original AccessDOS features built into the operating system as the "Accessibility Options" control panel. The functionality includes all of the configuration flexibility available in AccessDOS. In general, setting choices are given in seconds, and users can choose from four or five different settings for features such as the key acceptance delay or the debounce time. In the "Accessibility Options" control panel, *Slow Keys*, *Repeat Keys* and *Bounce Keys* are grouped together under the heading *Filter Keys*. The *Filter Keys* settings can be activated and deactivated using the keyboard shortcut of holding down the *Right Shift* key for eight seconds. Within *Filter Keys*, users can either activate *Bounce Keys* or both of *Repeat Keys* and *Slow Keys*.

The functionality of *Repeat Keys* is also accessible through the "Keyboard" control panel. Four repeat delay settings ("long" to "short") and over twenty repeat rate settings ("slow" to "fast") are available here. In the "Accessibility Options" control panel, five repeat delay settings are available, and these are quantified as 0.3, 0.7, 1.0, 1.5 and 2 seconds. There is also the option of suppressing all repeats. The repeat rate options are also different: settings of 0.3, 0.5, 0.7, 1.0, 1.5 and 2 seconds are available. It is not made clear how these relate to the range of settings available in the "Keyboard" control panel, except that the "Accessibility Options" settings override those of the "Keyboard" control panel when they are activated.

It is also possible to adjust the mouse speed (seven settings, "slow" to "fast") and double click speed (a sliding scale from "slow" to "fast"), as on the Macintosh. This is done in the "Mouse" control panel.

2.6.5 UNIX/X Windows

In October 1992, the Disabilities Action Committee for X (DACX) was formed (Novak and Vanderheiden, 1993; Walker et al., 1993). This group, co-ordinated by the Trace R&D Center, consisted of researchers and companies interested in developing access solutions for X Windows®. These solutions were intended to extend the existing facilities for PC and Macintosh platforms to UNIX workstations running X Windows. As a result, the AccessX package, compatible with SUN and Digital™ workstations, was created. AccessX includes all of the functionality of AccessDOS, and is available for many UNIX workstations including Solaris 2.4 and above, Digital VMS™ 6.0, Digital UNIX 3.2 and above, and OpenWindows™ 3.4 or above.

2.6.6 Summary of Software Configuration Facilities

In the main, the facilities provided on different platforms are very similar. The Macintosh has fewer facilities and fewer opportunities to control the behaviour of the facilities, while Windows 95 has a large and complex set of options. Between platforms, and within Windows 95, choices of setting for facilities such as the key repeat delay differ. Sometimes they are presented quantitatively, sometimes qualitatively. This makes it difficult to transfer knowledge between platforms.

The following section discusses the need for these facilities, and summarises existing research examining their adequacy.

2.7 Related research on Keyboard and Mouse Use and Configuration

The previous sections have described a large number of support tools available to keyboard and mouse users with motor disabilities. It appears from these facilities that input errors are a major source of difficulty, and that some errors are particularly frequently encountered. While Sections 2.1 and 2.2 outlined a large body of research describing the use of keyboards and mice by non-disabled users, there have been very few empirical studies of users with disabilities. August and Weiss (1992) present a summary of human factors literature as it pertains to the prescription and customisation

of input devices for people with motor disabilities. They examine aspects such as digit loading, digit travel and device positioning. They provide three case studies in which human factors research usefully informed device prescription, but warn that, in general, the results of studies with non-disabled subjects are not necessarily applicable to people with disabilities. As a result specific studies of keyboard and mouse usage by people with disabilities are very important. This section reviews the available research describing the input difficulties encountered by keyboard and mouse users with motor disabilities, and the use of the facilities described in Sections 2.5 and 2.6 to overcome these difficulties.

A number of references to specific difficulties related to disability can be found. Brown (1992) describes difficulties in performing multiple key presses that can render a program "virtually inaccessible", observes that unwanted characters generated by key repeats can present a "serious obstacle" to accessibility, and mentions that elimination of characters inserted by unintentional key activation can occupy "a great deal of time". Edwards (1995) states that hand tremor can make it impossible to strike keys without also hitting adjacent ones, and also mentions difficulties with key repeats. Vanderheiden (1992), and Poulson, Ashby and Richardson (1996) report the same difficulties. Vanderheiden differentiates between unwanted characters produced through accidental bumping of keys, and those caused by reactivation of a key after it has been pressed (bounce errors). He also mentions that pointing devices can be difficult or unusable for some people.

Brownlow et al. (1989) provide more detail of difficulties experienced by people with physical disabilities using pointing devices. Of the mouse, they report that physical disabilities can result in difficulty in grasping the mouse, accurately pointing to a target, clicking the mouse button without moving the mouse, and holding down the mouse button while positioning the mouse (particularly if different body parts are used for moving the mouse and holding down the button). They also report that the large range of movement required can be difficult and tiring to achieve, and that the mouse can be jerked out of range. Riviere and Thakor (1996) investigated the effects of age and disability on a tracking task using a mouse, comparing a group of 8 young, non-disabled adults, a group of 4 older subjects (aged 70 to 74 years), and a group of 5 subjects with disabilities, at least 4 of whom suffered from tremor. They concluded that mouse usage becomes increasingly inaccurate and non-linear with advanced age and disability.

While these reports provide a good overview of the types of difficulty frequently encountered, more detailed information, and information on the adequacy of the existing support facilities, is less easy to find.

One potential source of such data might be those organisations offering assessment of the assistive technology requirements of people with motor disabilities. Professionals working in organisations such as the Trace R&D Centre (Wisconsin, USA), the Hugh MacMillan Medical Centre (Toronto, Canada), the Foundation for Communication for the Disabled (UK), and KeyComm (Edinburgh, Scotland) regularly assess the suitability of different input devices and configurations for disabled individuals. While there is no standard assessment test, guidelines for assessment such as Lee and Thomas (1990) or Broadbent and Curran (1992) provide detailed instructions, including subjective assessment of typing speed and, more importantly, accuracy. Some software packages for assessment of skills such as cursor control (Brownlow et al. 1990; Schwartz and Milchus, 1992) have also been developed. However, there is still little empirically derived data to help choose an appropriate cursor control device for an individual (Casali, 1995).

Kondraske's (1988) observation that assessment is generally approached in "a pseudo-systematic and subjective" (Kondraske, 1988, p14) manner remains true. As DeCoste points out, "AAC users generally require adapted test methods using nonstandardized presentations due to time limits, limited speaking skills, or lack of fine motor skills" (DeCoste, 1997, p250). Given the great variation in the abilities and requirements of individuals to be assessed, the bewildering range of options available for computer modification, and the physically demanding nature of the assessment itself, it is hardly surprising that an individualised approach has been necessary. The goal of the assessment is to serve the client, rather than to investigate the adequacy of the input devices available as options. Detailed data on the use of the devices is rarely recorded.

Similarly, the software configuration facilities described previously were largely based on the experience of staff at the Trace R&D Center, and others working in the field. Some user testing was carried out in order to ascertain appropriate parameters, but again almost no empirical data are available (Gregg Vanderheiden, Trace Center Director, personal communication). An exception is the study reported by Lee and Vanderheiden (1987) investigating appropriate maximum speed of pointer movement in the *Mouse Keys* utility. Lee and Vanderheiden describe the accuracy and speed of pointer positioning achieved by seven subjects using *Mouse Keys*. They found that

factors such as the match between the subject's ability and the available key repeat settings, and subjects' ability to predict pointer motion were also important.

It seems, then, that while there is much awareness of the difficulties that keyboards and mice can pose for people with motor disabilities, formal studies of these difficulties have not been carried out. Similarly, facilities designed to overcome these difficulties have evolved through experience and informal experimentation, rather than formal evaluation. One reason for this is the enormous variability in the requirements of individuals with disabilities, which makes it difficult, if not impossible, to recruit a representative sample of users for any proposed new facility. Nevertheless, as these facilities are becoming more and more standardised between operating systems, some attempt at evaluation of their adequacy and usability would be useful. This topic will be returned to in Chapters 5 and 6.

Examination of the existing literature, the configuration facilities available, and their advertised benefits, reveals that input errors are a particularly common manifestation of input difficulties, and one for which software and hardware support can offer solutions. It is these input errors with which this thesis is primarily concerned. The following section examines these errors more closely, and defines the term *performance errors*, which will be used to describe them throughout this thesis.

2.8 Performance Errors and Other Error Classifications

Errors in human-computer interaction can be considered to lie on a spectrum ranging from high-level cognitive errors (e.g. errors in the user's goals and plans) down to low level errors in the execution of a plan.

Many researchers have attempted to produce a classification of human errors. One major classification, proposed by Norman (1981), divides errors into mistakes and slips. A mistake is an error in the user's specification of a desired action: a planning failure. A slip is "a form of human error defined to be the performance of an action that was not what was intended": an execution failure (Reason, 1990).

On the surface, it may appear that performance errors could be classified as slips, since they are indeed failures in the execution of a desired action. However, the sense of

Norman's definition is not that the desired action was performed inaccurately, but that an accurate, but wrong, action was substituted for the desired one.

A typical example of a slip would be for someone to pour their breakfast cereal into the coffee cup instead of the bowl. Slips are accurate, deliberate movements, not the unintentional movements produced by a physical disability, or the inaccurate movements of a careless typist.

Norman's classification, and most other classification schemes (e.g. Egan, 1988), only cover cognitive errors. One exception is the classification proposed by Swain (Miller and Swain, 1987). He takes a system oriented approach, and looks at human error in terms of incorrect system inputs, or human outputs. Two broad classes of error are defined:

- Errors of Omission: A required action is not performed.
- Errors of Commission: The user performs an incorrect action, fails to meet time constraints, or makes a wrong estimate of the required parameters for an action.

Unlike Norman's, this classification has the advantage that from the system viewpoint the same error is always classified the same way across different systems and users. Swain's classification does include accidental typing errors, but it does not distinguish them from cognitive or other causes of error, e.g. mechanical failure in the input device.

Gentner et al. (1983) provide a classification of typing errors made by touch typists performing a transcription task. Their approach uses a <hand, finger, position> notation, in which errors could occur in the typist's specification of hand, finger, or finger position. In addition to errors in these aspects, they define insertion, transposition, migration and interchange errors, in which letters are inserted, swapped with immediate neighbours, moved to a different position in the text, or swapped with non-neighbouring letters, respectively. Other error classes include doubling the wrong letter in a word and omitting letters. Gentner et al. also define 'misstrokes', which "can be traced to inaccurate motion of the finger, as when one finger strikes two keys simultaneously, or contacts another key in passing with sufficient force to activate it" (Gentner et al., 1983, p41).

Further discussion of error classifications with respect to computer use can be found in Chapter 10 of Baecker et al. (1995).

This thesis is concerned with performance errors, which are defined as:

> *errors attributable to physical inaccuracy in the manipulation of input devices.*

This class of errors includes the misstrokes described by Gentner et al., and other error types such as:

- failure to press a key hard enough;
- pressing a key for too long, producing unwanted repeat copies;
- slipping off a key, perhaps activating nearby keys;
- missing a key;
- accidentally pressing the mouse button;
- inaccurate positioning of the mouse.

There has been little research into performance errors. Chapter 4 will present some more specific examples of the performance errors observed in an empirical study of keyboard and mouse use. While all users, disabled or not, occasionally make such errors, for some they can be a major problem, to the extent that they render the input device unusable. For example, if a user presses keys for a length of time such that most key presses produce two characters, then a great deal of time could be spent typing two characters, and then pressing *Delete* and erasing both of them, leaving the user no further forward.

In comparative evaluation of input devices, keystroke accuracy or pointing accuracy are often used as performance measures (Hurlburt and Ottenbacher, 1992). For example, keystroke accuracy can be defined as the number of correct characters divided by the total number of characters typed. Using this measure, the fewer errors, the better the performance. Similarly, MacKenzie, Sellen and Buxton (1991) studied the performance of a mouse, trackerball and graphics tablet for both pointing and dragging tasks, and used error rates as one of the major performance measurements. MacKenzie et al. note the presence of "dropping errors" when dragging - users sometimes unintentionally released the mouse button before reaching their target. Such errors are another example of performance errors. However, this investigation focused on "normal motor variability", and these dropping errors were excluded from their analysis.

Grudin (1983) describes research into error patterns in transcription typing, using both novice and skilled typists. In this and related work (see Grudin, 1983 for references), errors are seen as an important source of information about the development and organisation of this complex motor skill. While misstrokes are less interesting to Grudin than the other error classes within the classification scheme presented by Gentner et al., he does observe that the most frequent errors for experts were insertion errors, and the overwhelming majority of these appeared upon video inspection to be misstrokes.

Similarly, Peterson (1980) reports that "typographical errors of typing", termed "keyboarding errors", are an important and significant source of errors, particularly in large databases. Although Peterson does not give a precise definition of keyboarding errors, the impression gained is that they are strongly related to misstrokes and performance errors.

Errors such as misstrokes are rarely studied in detail. Historically, HCI research has either examined expert, error-free performance (e.g. Card et al., 1987; Roberts and Moran, 1983), or concentrated on cognitive errors and their causes (e.g. Egan 1988; Miller and Swain, 1987; Norman, 1983).

This may be because performance errors are generally considered a relatively infrequent occurrence for the average, non-disabled computer user (Bailey, 1983), and can be immediately corrected. Cognitive errors, on the other hand, will have a major effect on the success of the interaction with the application. For example, one case study described by Baecker and Buxton (1987), an evaluation of text editors and word processors, only includes errors which took at least fifteen seconds to correct in the estimate of time spent making and correcting errors. The majority of performance errors would therefore be excluded. Another reason for the focus on cognitive errors is that, while occasional performance errors are considered inevitable, cognitive errors can often be remediated by improved interface design (Norman, 1983). It is only when performance errors become a major source of difficulty that they become interesting, and investigation of such cases is often considered something of a specialist area, outside the bounds of mainstream human-computer interaction (Newell, 1995).

2.9 Assessment and Support

Casali (1995) observes that it is increasingly difficult for clinicians to keep pace with the proliferation of new input devices and other support mechanisms for computer users with disabilities. Whereas in the past, assessment largely consisted of a trial-and-error approach, with assessment centres stocking examples of different devices for clients to try, today this approach is becoming less feasible. Casali argues that a more principled approach, based on explicit specification of the physical skills and effort required to operate different devices, and the assessment of users' abilities with respect to those skills would allow clinicians to home in on appropriate devices among the many available.

Once an appropriate device is chosen, the best possible configuration for the individual's requirements must be found. Many authors emphasise the need for systems to be configured to suit individual users with disabilities (Baecker and Buxton, 1987, p675; Krogh Hansen and Wanner, 1993; Poulson, Ashby and Richardson, 1996, 5.3.8; Shein et al., 1989a, 1989b).

Whilst configuration options for keyboards and mice have already been described, other input devices can also be configured. For example, switch users have a choice of different scanning speeds, and configuration consists of finding the fastest speed which does not compromise the accuracy with which the user can make selections.

One very important aspect of technology use by people with motor disabilities is the follow-up training and support provided (Cox, 1996; Watkins, 1989). Unfortunately, training is often limited by the availability of the trainer (Broadbent and Curran, 1992).

For a keyboard or mouse user with motor disabilities, formal assessment is one way of discovering what configuration facilities exist, and which of these may be relevant. Those who do not have formal assessment may have a teacher who is helping them to learn to use the computer, and can find out about configuration facilities through this teacher. However, if the student is using the computer in an educational setting, their teacher may not be knowledgeable about such facilities. Stevenson College in Edinburgh has recently completed an examination of the use of technology in education, and found that while staff were keen to exploit the benefits that technology could provide for students with disabilities, they had only a limited amount of time to devote to learning about such technologies themselves. One teacher reports:

> Once I discovered how to use the special software, it was amazing how well it enabled the student to keep up in class. It was a bit embarrassing that she knew more than I did about it at first! (Cox, 1996, p24)

Other people may be using computers at home, and be reliant on friends and family members, or manuals and on-line help systems for support and information. Manuals can be physically difficult for some people to manipulate, while many users find on-line help systems difficult to use (Sellen and Nicol, 1995). They also require the user to be proactive in searching for solutions to difficulties. There is, therefore, a danger that those who could benefit from the existing configuration facilities are not always aware of their existence. Chapter 6 provides evidence that there is indeed a lack of awareness of configuration facilities among keyboard users with motor disabilities.

2.10 Summary

It seems, then, that while the development of the typewriter was to some extent spurred on by its potential to allow some people with motor disabilities to produce writing, today's keyboard designs are the result of many conflicting influences, and even evaluation of their use by non-disabled people is lacking in some areas. As a result, they are not as easy to use as they could be. Cursor control devices, including the mouse, have also been developed for and largely evaluated with expert users. Comparisons of these devices have produced contradictory results, and it appears that the ideal device is strongly dependent on the skills and preferences of a particular user, and the demands of their task.

For people who have difficulty using mice and/or keyboards, there are many alternative input devices available, including switches, eye gaze tracking, voice recognition and alternative keyboards. Choosing an appropriate input mechanism involves a trade-off between several often conflicting factors, over and above the physical ease of use and achievable accuracy of the chosen method. Some devices do not allow access to the full functionality of all mainstream applications, some are slow, others expensive. Users may also have strong personal preferences unrelated to their performance with the device itself.

For those who choose to use the standard keyboard and/or mouse, there are many modifications that can improve their accessibility. These range from external facilities

such as arm and wrist rests, keyguards and key latches to software which alters the behaviour of the system. Included in the latter class are macros, accelerated writing systems and device configuration options. These configuration options are a complementary technique to macros and accelerated writing systems - the latter reduce the amount of input required, while the former relax the constraints on physical accuracy required by the devices.

Configuration facilities include *Sticky Keys*, *Repeat Keys*, *Bounce Keys*, *Slow Keys*, and *Mouse Keys*, which have been described in full. These originated as third-party software patches but are now available in similar forms on the majority of modern operating systems. Appropriate configuration is crucial for many people, yet some users, particularly novices, remain unaware of potentially useful facilities.

A key point that has emerged is the lack of formal research describing the difficulties people with motor disabilities can experience when using different input devices. Chapter 4 will provide some empirical data describing such difficulties. While the utility of the software configuration facilities is not in doubt, there have been no published studies describing their use, as far as the author is aware. In Chapter 5 of this thesis, some detailed usage data for *Sticky Keys* and *Repeat Keys* is presented, while Chapter 6 will compare the difficulties observed in Chapter 4 with the available facilities, and discuss the extent to which the common difficulties are catered for, making suggestions for extensions to the current provision.

Chapter 3

Adaptive Interfaces and User Modelling

Thimbleby (1995) discusses the frequency with which poor system design forces users to adapt their behaviour to match the requirements of a system, work around bugs, and avoid difficult, confusing or incomprehensible functions. While many users do find ways to work around system design flaws, this is not possible where the system requires some cognitive or physical ability that the user does not have. In such cases, access is not possible unless the interface can adapt or be adapted to match the user's requirements.

Chapter 2 has described difficulties that people with motor disabilities can have with keyboards and mice, and a number of software configuration mechanisms allowing users to surmount these difficulties. In this respect, existing interfaces can be said to be *adaptable*: open to modification to suit individuals with different requirements.

In order to take advantage of these facilities, the user, or some other human agent, is currently required to explicitly adapt the interface of the system. This requires knowledge of the facilities available, appropriate settings for these facilities, and the mechanism for performing the adaptation.

Many users may not know about potentially useful configuration facilities, many may also be unable to activate the facilities themselves. That some users do have physical difficulty in activating access facilities is acknowledged by the inclusion of the emergency activation feature in AccessDOS, as described in Chapter 2. This feature was designed to help users who do not have enough control over the keyboard with the default configuration to be able to choose and activate the settings they required.

As mentioned previously, knowledgeable human support is not always available. Having an adaptable system may, therefore, not be enough to ensure that all users have

as high a level of access as possible. To bridge this gap, some responsibility for system adaptation could be shouldered by the system itself, providing an *adaptive* interface.

Much research, particularly in human-computer interaction, is currently devoted to systems which can adapt to different users (McTear, 1993). Examples include information presentation systems which adapt to match a user's goals or experience (e.g. Linden, Hanks and Lesh, 1997; Meyer, 1994), tutoring systems which adapt teaching strategies to address a user's particular difficulties and proficiencies (e.g. Beck, Stern and Woolf, 1997), and application program interfaces which adapt to the way in which they are used in order to save the user time and effort (e.g. Cypher, 1991; Debevc et al., 1996; Greenberg and Witten, 1985; Lerner, 1992).

Any system which uses knowledge about a user to guide an adaptation mechanism must possess some form of model of that user. An active research area in its own right, user modelling has a long history as an artificial intelligence technique for the guidance of intelligent teaching systems (see Polson and Richardson, 1988; Sleeman and Brown, 1982; and Wenger, 1987 for summaries of early research in this field), and it has grown to encompass many different methods and approaches.

This chapter examines research in adaptive interfaces, and how it could be applied to the configuration of input device behaviours. Section 3.1 discusses existing forms of adaptive interface. An overview of user modelling, and available user modelling techniques, follows in Section 3.2. Finally, Section 3.3 outlines a proposed *adaptive configuration support tool* based on a model of the current user's input abilities. This provides the background against which the user model to be described in Chapter 7 has been developed.

3.1 Adaptive Interfaces

Adaptive and adaptable interfaces are of increasing interest as applications become more complex and a wider variety of computer users emerge. They offer a mechanism whereby very different users can be accommodated by a single application program. Similarly, a user whose needs change over time can also be accommodated. Such changes may be gradual, as a user becomes more experienced, or sudden, as they switch between different tasks.

Offsetting these potential advantages, researchers have identified a number of potential pitfalls. If users are to perform their own customisation this requires them to learn skills and expend time and effort on a task tangential to the task for which they are using the computer in the first place. This additional effort may not be acceptable to users (Benyon and Murray, 1993a; Innocent, 1982; Mason, 1986). If the system performs the customisation automatically then users may become disorientated or confused - just as they are beginning to understand the system, it begins to behave differently (Innocent, 1982; Norcio and Stanley, 1989). It is commonly observed that users like to feel in control of a system, and self-adaptation may undermine that impression (Cypher, 1991; Kühme, 1993; Meyer, 1994; Norcio and Stanley, 1989). As the system adapts to the user, and the user adapts to the changes in the system, it is possible that this cycle will never stabilise (Innocent, 1982) - this process has been dubbed 'hunting' (Browne, 1990).

Between the extremes of user and system adaptation are a number of intermediate positions representing hybrid approaches. The following section presents a classification of adaptive systems which illustrates these different approaches. In Section 3.1.2 the advantages and limitations of the most relevant approaches will be illustrated using examples from the research literature.

3.1.1 A Classification of Adaptive Systems

Interface adaptation is one of a number of possible adaptations a system could make. For example, adaptations can also be made to information presented, or system responses to given commands. While the following is a broader discussion of adaptive systems in general, the conclusions can be directly applied to interface adaptation.

Dieterich, Malinowski, Kühme and Schneider-Hufschmidt (1993) identify four stages in the adaptation process:

1. *Initiative*: A decision to perform adaptation is made.
2. *Proposal*: One or more specific adaptations are proposed.
3. *Decision*: An adaptation is chosen.
4. *Execution*: The chosen adaptation is executed.

Given a user and a system, each stage could be performed by either party. Dieterich et al. (1993) use their four stage view as the basis for a flexible classification of adaptive systems. They describe six interesting classes of adaptive behaviour, based on particular combinations of user and system responsibility for the four stages. These are given below. Note that under Dieterich et al.'s scheme, 'self' refers to the system, rather than the user.

- *Self-Adaptation*: The system initiates, proposes, decides and executes adaptations without any contribution from the user.
- *User-Initiated Self-Adaptation*: The user initiates the process of adaptation. The system proposes, decides on, and executes the adaptation.
- *User-Controlled Self-Adaptation*: The system initiates adaptation and presents proposal(s) to the user, who makes the final decision. The system then executes the chosen adaptation.
- *Computer-Aided Adaptation*: The user initiates the adaptation process, proposals are made by the system, decided on by the user, and executed by the system.
- *System-Initiated Adaptation*: The system informs the user when it might be reasonable to perform adaptation, while it is left to the user to devise and choose the adaptations. Execution could be shared between the system and the user.
- *Adaptation*: The user performs all four stages. The system may provide some support in the execution of the chosen adaptations. This class is equivalent to the adaptable systems mentioned in the introduction to this chapter.

Dieterich et al. (1993) go on to use this scheme to classify existing adaptive systems, some of which use more than one strategy. Many adaptive systems also include a fifth stage - *evaluation* - in which the success of the implemented adaptation is evaluated. This is particularly important when the decision to implement an adaptation was not made by the user.

A number of adaptive systems, their place within this classification, and the advantages and limitations of each approach will be discussed in the following section.

3.1.2 Examples of Existing Systems

As discussed earlier in this chapter, some users in this domain may have difficulty in performing adaptation themselves. The most relevant approaches are therefore those in which the system implements adaptations. This section presents examples of systems from three of the four relevant classes described by Dieterich et al. (1993): self-adaptive, user-controlled self-adaptive and computer-aided adaptive. These examples will be used to illustrate the advantages and disadvantages of each approach. Summaries of other adaptive systems research can be found in Benyon and Murray (1993b), Schneider-Hufschmidt, Kühme and Malinowski (1993) and Brusilovski (1996). Examples of user-initiated self-adaptation are not given due to a lack of clear examples of systems in which users initiate adaptation without knowing what adaptations will be made by the system. Intelligent tutoring systems which ask users their preferred interaction style, and then adapt the tutorial according to the user's preference are perhaps the closest examples. Users do not know what adaptations are being made. However, even here, the system initiates the adaptation by asking users their preferred style, and the user makes the style decision.

Self-Adaptive Systems

In self-adaptive systems, the user has no control over the adaptation process. For example, *Dialog* is an adaptive interface to a statistical tool (Maskery, 1985) in which the system adapts the style of the dialogue with the user, graduating from a system-led, system-controlled interaction to a user-led interaction, and forcing users to learn a new approach to the system. In an experimental study of naïve users of *Dialog*, most had difficulty making this transition. Meyer (1994) suggests that these negative reactions were due to the user being forced to adapt to changes in a previously learned stimulus-response relationship. She describes a further study by Hockley (1986)(cited in Meyer, 1994) in which a command-based interface adapted the level of feedback and prompting it supplied to users, and was not well received. She argues that the differences between levels may have interfered with previously learned command sequences.

Another self-adaptive system which improved user performance, but was unpopular with its users is presented by Morris, Rouse and Ward (1988). In this study, users performed two tasks simultaneously - a manual tracking task and a visual search task

which involved recognising boats of a certain shape while scanning over a waterway scene. An aid was present, which could provide users with assistance in the visual search task. Over some classes of waterway scene, the aid was more effective than the user, while in other classes of scene users tended to perform better. In the adaptive version of the system, allocation of search tasks between the user and the aid was controlled by the system. In the manually controlled version, users chose when to use the aid and when to perform the search task themselves. Overall performance was better when the system controlled task allocation, but users preferred the manually controlled condition, and often cited a desire to be in control as their reason for this preference.

Meyer adopted a more successful approach in her Retail User Assistant (RUA) (Meyer, 1994). This system adapted the level of help provided, according to the user's experience and skills. Here, the form of user interaction with the system was not changed by the adaptations, and users responded very positively.

In one early study, a personalised menu system for dictionary lookup was compared to a static (alphabetically structured) menu system (Greenberg and Witten, 1985). The adaptive system altered the menu structure so that commonly requested entries were accessed more easily. The adaptive system was found to be faster, fewer errors were made using it, and most subjects preferred it. However, Greenberg and Witten studied the adaptive system in its equilibrium state, so that the system had already largely adapted to the frequencies in the data before the subjects were introduced. It is not clear whether the system was still adapting at all while the subjects used it. While the result of the adaptation seems to have been beneficial, the effect of dynamically adapting the menus is not known.

Another form of self-adaptation is the automatic customisation of structure editors. A structure editor provides support for writing and editing computer programs, by providing macros which embody the common constructs of a programming language. Lantern (Lerner, 1992) is a system which monitors the way in which the default macros are extended by users, and incorporates common patterns in the user's behaviour into the existing macros. It evaluates the success of these customisations by monitoring whether they are retained or deleted by the user. The results of a small field test were mixed. One user, out of a total of five, rejected the system at an early stage, but two preferred the adaptive editor to the original. The opinion of the other two users is not reported.

These examples illustrate that while self-adaptive systems do free the user from having to learn configuration skills, fully automated adaptation can be disorientating and unpopular with users. Adaptations which force the user to change the way they use a system appear to be unacceptable, while those such as RUA, whose adaptations do not affect the input required of the user, are more likely to be accepted. In an adaptive interface it may be important to ensure that users are not forced to abandon actions they have already learned to perform.

User-Controlled Self-Adaptation

In user-controlled self-adaptation, the user makes adaptation decisions, but the system initiates changes, and proposes and executes adaptations.

One example of this type of adaptation is the creation of macros which embody repetitive sequences of actions. One such system - EAGER (Cypher, 1991) - detects regular patterns in the user's activity and generalises these into a program. The user may then ask EAGER to complete the task automatically. EAGER was successfully used by a small set of users, largely without instruction. The adaptation does not force any changes to the way the user interacts with the system, and does not require any explicit actions to define the macros to be generated.

In this approach, adaptations have to be explained to users. While users must understand the adaptation suggestions presented, they have little additional work to perform. Users are in control of the adaptations made. One potential difficulty with this approach is that the user must somehow be informed when the system has a suggestion for adaptation, preferably in a way that does not interrupt their work. EAGER tackles this problem by showing an icon on the screen, representing a newly generated macro, and highlighting the actions it predicts that the user will perform next, rather than interrupting the user. When the user is satisfied that the highlighted actions are correct, they click on the icon to activate the macro. Stepping and undo functions are also provided, to improve user control over the adaptations.

Computer-Aided Adaptation

In computer-aided adaptation, the user controls both the timing and content of adaptations, but does not have to devise or execute them.

Debevc et al. (1996) describe an adaptive toolbar for Microsoft Word for Windows of this type. The toolbar contains icons representing the user's most frequently (and recently) used commands. The system proposes changes to the set of icons represented on the bar, and the user can view the suggestion and choose whether the system should implement it or not. The availability of a suggestion for adaptation is indicated by a tone, and a change of colour of the adaptive bar itself. While this action reduces the extent to which the user can be said to initiate adaptation, nevertheless it is the user who chooses when to view the system's suggestions.

Debevc et al. evaluated their system with both novice and expert users of Word In the group using the adaptive toolbar, experts and novices created and used similar numbers of icons. In the non-adaptive group, experts created and used more icons than they did with the adaptive toolbar, and novices created no icons at all. Users generally reacted favourably to the adaptive bar. These results imply that the adaptive bar did make it easier for novices to perform toolbar adaptations.

Computer-aided adaptation does not interrupt users when they are performing tasks. While it does require users to be aware that adaptation is possible, and know how to initiate it, users do not need to devise their own adaptations. This approach appears to offer a good trade-off between user control and system support for adaptation.

3.1.3 The Implications of Research Results for Input Device Configuration

Dieterich et al. (1993) conclude that much work on self-adaptation has failed to show that the adaptations made really matched the user's needs, and that the computer-supported adaptation and user-controlled-/initiated self-adaptation approaches, which give the user more control, are a more promising research area. EAGER (Cypher, 1991), for example, was able to create useful macros, and was used successfully even without instruction.

Other authors (e.g. Benyon and Murray, 1993a; Kühme, 1993) have also noted that research into self-adaptive systems has not always fulfilled expectations. One reason

for this is that because the user has no control over, and may have no understanding of, the adaptation, there is a danger that the system may seem to be unpredictable and therefore unreliable and more difficult to learn and use.

In the context of input device configuration, where users may be unaware of the configuration options open to them, a system initiated approach is required. Computer-aided adaptation is therefore inappropriate. The system must also make proposals for adaptation, for the same reason. Self-adaptation or user-controlled self-adaptation are the most promising approaches.

In spite of the negative research results reported for self-adaptation, it retains the advantages that the user is not required to understand the adaptive mechanism, or to learn any extra information in order to be able to use the system. Automatic adaptations can also be implemented seamlessly, without interrupting the user's task. If necessary, information about what adaptations have been made can be provided upon request from the user. Meyer's Retail User Assistant (Meyer, 1994) illustrates that self-adaptive systems can be successful if they do not force users to change their interaction style.

For some configuration facilities, self-adaptation may be the most appropriate approach, while for others a greater level of user involvement may be preferable. Consequently, both self-adaptive and user-controlled self-adaptive approaches will be considered in Section 3.3 where the possible form of an input device configuration support system is discussed in more detail.

Both of these approaches require a system capable of modelling a user's configuration requirements. The following section provides an overview of the field of user modelling, and user modelling techniques that have proved effective.

3.2 User Modelling

User modelling has emerged as a useful technique in a number of different areas. It has been used, for example, to model the knowledge and misconceptions of students in intelligent tutoring systems (Corbett and Bhatnagar, 1997; McCoy and Suri, 1996; VanLehn, 1988), and to infer the goals and interests of users searching for information on the Internet or in large databases (Ambrosini, Cirillo and Micarelli, 1997; Linden,

Hanks and Lesh, 1997), as well as to decide the preferences and experience of users of adaptive interfaces (Benyon, 1985; Greenberg and Witten, 1985; Lerner, 1992; Mason, 1986; Meyer, 1994).

As a mechanism for guiding interface adaptation, user models could play an important role in accessibility of systems and applications at a number of levels. For example, McMillan (1992) suggests incorporating specific models of different subsets of users with special needs into standards for off-the-shelf computing interfaces. User models have also been used in specific assistive technology systems. Accelerated writing systems such as PAL (Newell et al., 1995) and Messenger (Venkatagiri, 1993), record information about a user's frequently used words in order to improve the efficiency of word prediction. Alm, Arnott and Newell (1992) describe CHAT, an augmentative communication system which uses information about the user's moods, in addition to a model of conversation, in order to make appropriate suggestions as to the user's next contribution to the conversation. Gutkauf, Thies and Domik (1997) describe a user model incorporating information about abilities such as colour perception, used to adapt the presentation of colour charts in literature published on the Web.

User modelling is in general a promising approach for improving interface and system accessibility for people with physical and other disabilities.

3.2.1 User Modelling and When it is Appropriate

There are many possible definitions of what constitutes a user model. For example:

> "the knowledge and inference mechanism which differentiates the interaction across individuals." (Allen, 1990, p513)
>
> "a system knowledge source that contains explicit assumptions on all aspects of the user that may be relevant for the dialog behavior of the system." (Kass and Finin, 1989, p83)

Both of these definitions focus on explicit models which are actively used by a system to adapt the method of interaction with, or information presented to, a given user. While this characterisation adequately describes the model presented in this thesis, the scope of user models can in fact be extended beyond this. For example, Bull (1997) describes a system in which the user model is primarily a vehicle to promote student

awareness of, and reflection on, their own performance in foreign language learning. Similarly, the model of configuration requirements described in this thesis could provide information directly to a user, or to a professional concerned with establishing the ideal configuration for a user, rather than forming the basis of an adaptive system.

Both Rich (1983) and Kass and Finin (1988) define some characteristics of systems for which individualised user models are appropriate. These include:

- Systems with a diverse class of potential users (Kass and Finin, 1988; Rich, 1983) - a single form of system interaction is unlikely to be appropriate for all users, and may be very inappropriate for some.
- Systems that are required to adapt to individual users (Kass and Finin, 1988; Rich, 1983) - if the task of the system is to cater to different users, for example helping them to find interesting web pages (Ambrosini et al., 1997) or making book recommendations (Rich, 1983), then some model of the user's interests will be helpful.
- Systems which take responsibility for successful system-user communication (Kass and Finin, 1988). For example, consider a simple command system. If the user mistypes a command, it will be rejected - the user is responsible for specifying requirements in terms that the system can interpret. An alternative version of the command system, upon receiving a mistyped command, could attempt to infer the user's intended command, confirm with the user, then carry out the command. Such a system is shouldering greater responsibility for the success of the interaction, and a model of a user's previous inputs is likely to be helpful in interpreting an unrecognised or ambiguous command.

3.2.2 Types of User Model

User models come in many forms, and can be applied in a multitude of ways. The goal of using a model is most often to make a system more acceptable to its users. Many researchers have attempted to classify the important dimensions of user models. This section examines the primary dimensions.

Canonical vs. Individual Models

Rich (1983) observed that the majority of systems are designed with a *canonical* user model, which represents a typical or average user. The model is often built into the system, and unalterable. For example, a keyboard is designed so that the size of keys and spacing between them are acceptable to the average person. The model of the user is embodied in the final keyboard, and cannot be changed. Canonical user models are most applicable in systems such as hardware devices, where physical features of the design cannot be dynamically altered.

The disadvantage of such systems is that their canonical model may not correspond to any real user, and the diversity in real users may mean that the chosen design is difficult or impossible for some people to use. For example, a system which can only present information graphically is unsuitable for blind users.

Adaptive systems rely on *individual* user models to allow them to dynamically alter aspects of the interface or system behaviour. An individual user model contains information recorded by a system about the current user, which can be used to customise the system. While individual user models enable highly flexible and more usable interfaces, accessible to a wide range of people, there is a cost associated with designing such systems. Not one but several interface styles may be required. In addition, the user model must somehow be created dynamically.

Explicit vs. Implicit Modelling

Rich (1983) identifies two mechanisms for acquisition of user models. The first is that users specify their own requirements or preferences *explicitly*, perhaps even performing their own customisation. This avoids the danger of a system making inappropriate guesses in situations where the user can specify their own needs. However, this specification in itself may demand a non-trivial level of expertise, and novice or occasional users may find this difficult. Users are not always able to provide reliable information about their own requirements (Kühme, 1993).

A second difficulty with explicit modelling is that users are required to spend time specifying the contents of the model, either by answering questions, or actively specifying requirements. This task is tangential to their reason for using the system in

the first place, and many researchers (Innocent, 1982; Mason, 1986; Rich, 1983) have argued that the time and effort required may not be acceptable.

An alternative is to use an *implicit* modelling technique, in which the model is built up from inferences made from users' actions in using the system. While this introduces difficulties in ensuring the accuracy of the model, it allows users to use the system in a natural way, and does not require them to learn any additional skills or spend time setting up the system.

Static vs. Dynamic Models

Kass and Finin (1989) argue that Rich's (1983) classification confuses the method of model acquisition (explicit vs. implicit) with the degree of modifiability of the model. They introduce degree of modifiability explicitly as 'dynamic vs. static' models. A *static* model does not change during a session, while *dynamic* models may be continuously updated. Explicitly constructed models are usually static, while implicitly built models may be dynamically updated during an interaction session.

Long- vs. Short-term Models

Rich (1983) defines a *long-term* model as one which includes long-term characteristics of users, such as areas in which they are expert. Conversely, a *short-term* model contains short-term information such as users' current goals and plans. Brajnik and Tasso (1994), on the other hand, define long-term models to be those with a lifetime of more than one session, regardless of the information contained therein. Whichever definition is used, long-term models require storage and recall between sessions.

Other dimensions

Several other researchers have proposed additional dimensions, such as the type of information modelled and the representation used (Allen 1990). Kass and Finin (1989) include the use of the model in their analysis. They define the method of use to range from *descriptive*, providing information about the user, to *prescriptive*, used to simulate or predict user actions.

VanLehn (1988) proposes a three-dimensional classification of student models used in intelligent tutoring systems. His classification is intended to describe systems which teach cognitive skills, and includes the information available about the student's mental states, the type of knowledge being taught, and the way in which differences between expert and student knowledge are represented.

Finlay (1990) provides a review of several user modelling taxonomies, and also argues that it is important to specify the aspects of the user being modelled, and the purpose (or use) of the model. She presents a framework of defining issues for user modelling, based around the agent creating and maintaining the model, the model itself, the information presented to the agent, and the use to which the model is put.

Jameson, Paris and Tasso (1997) offer a practical breakdown of the user models that were presented at the Sixth International User Modeling Conference. Their breakdown includes the purpose of modelling, content of the model, input to model construction, and methods of constructing and exploiting the models.

Construction of models and exploitation of models are strongly linked to the underlying representation used. The following section describes techniques that have proved effective in representing and using user models.

3.2.3 User Modelling Representation Techniques

In general, a user model is a description of the preferences, goals, skills or knowledge of a user. The way in which this information is represented depends very much on what exactly is being stored, and how it is to be used. In adaptive dialogue systems, for example, a user may simply be classified on a scale ranging from novice to expert, for each task within the given domain (e.g. Benyon, 1985; Maskery, 1985). For intelligent teaching systems, the model must represent what portion of a domain of knowledge the user is familiar with. It may also be necessary to represent the user's misconceptions or beliefs about the domain.

Many techniques for representing user models have been developed. This section describes some of these. The focus here is on symbolic user models, and those based on numerical probability-based techniques. User modelling using neural networks is beyond the scope of this thesis. See Chapter 9 for a discussion of the potential relevance of connectionist techniques to the modelling of input device usage.

The Overlay Model

A common approach to user modelling, at least within the context of tutoring systems, is to compare the current user's behaviour with that of an expert user. In the overlay model, a user is deemed to be expert when they display all of the behaviour that defines an expert. This approach is useful in instructional systems where the goal of the system is to impart a set of information to the user. One example is GUIDON, a system for teaching medical diagnosis (Clancey, 1987). The domain was represented by a set of rules in an expert system, and students were modelled by marking each rule according to whether or not they knew and could apply that rule. One limitation of this approach is the underlying assumption of consistency in the user's behaviour. There is no built-in mechanism for handling intermittent evidence.

Perturbation Models and Bug Libraries

Where it is necessary to represent incorrect as well as correct user behaviour, perturbation models or bug libraries have been successful (Holt et al., 1991; VanLehn, 1988). These techniques are similar to the overlay model, in that the user is represented with reference to an expert's behaviour, but deviations from that behaviour are also represented. Common deviations, or *bugs* can be specified and stored. These are then used to try to account for the user's behaviour. This approach has been applied to domains such as teaching arithmetic skill (Brown and Burton, 1978). Libraries of bugs can be obtained from sources such as educational literature, empirical data on student behaviour, and learning theories for the subject domain. It can, however, be difficult and time consuming to generate a reasonable and comprehensive library of bugs.

A variation on the use of bug libraries is to generate a library of *bug parts*, which can be combined dynamically to generate bugs which account for a user's behaviour (VanLehn, 1988). This technique has the potential advantage that a small bug part library can represent many bugs, so such libraries may be easier to construct. It also gives systems greater flexibility in modelling individual users.

As with overlay approaches, these approaches tend to assume consistency in the student's behaviour. In the arithmetic domain investigated by Brown and Burton (1978), there is evidence that students do not apply buggy procedures as consistently as might have been hoped (Self, 1988).

Stereotypes

Another approach to user modelling, pioneered by Rich, is to use stereotypes of different classes of user (Kay, 1994; Rich 1983, 1989). A stereotype is a collection of traits that commonly occur together. Using this approach, a system has a set of stereotypical users instead of a single domain model. A new user is classified according to which of these stereotypes best matches their behaviour, and stereotypes can be blended to produce an individualised model. A classic example of this approach is Grundy (Rich 1983), a system which recommends books that a user might like to read. System observations are used to trigger stereotypes by which a set of new user characteristics are inferred.

It has been suggested (McMillan 1992) that it may be possible to develop a set of stereotypes of people who have difficulties in the manipulation of computer input devices, and use these to customise user-system interactions. Such stereotypes could, in theory, be based on knowledge about a user's disability, or on inferences made from problems observed.

In the former case, for example, knowledge that a user had lost an arm as a result of an industrial accident could reasonably lead to the conclusion that they may be interested in using *Sticky Keys*, but that all other keyboard configuration facilities are unlikely to be relevant. Knowing that the user's disability was the result of a stroke, on the other hand, would provide little useful information, due to the enormous individual variation in the after-effects of a stroke.

Bayesian Networks

Probabilistic methods such as Bayesian networks are finding increasing popularity as user modelling techniques. Much of their appeal lies in their ability to handle many

sources of evidence, in order to draw inferences and make predictions about which there is some associated and measured uncertainty. Bayesian networks, or belief networks, consist of a graph representation of a potentially large set of variables, represented by nodes. These nodes are linked by directed causal connections, indicating the influence of nodes on each other. Each node has an associated conditional probability distribution, specifying the probability of a given value at a node for every possible set of values of nodes which influence it.

Bayesian networks have been used in problems such as the Lumiere project at Microsoft, which interprets a user's actions in order to provide appropriate help topics, and formed the basis for the Office Assistant shipped with Microsoft Office '97 (Horvitz, 1997). Jameson (1996) provides many other examples of the application of Bayesian networks to user modelling problems.

One limitation of Bayesian approaches is that they require an initial set of probabilities, which may be difficult to determine with any verifiable accuracy. They are most appropriate for large problems with many interlinked variables, and may be overly complex for applications with relatively few variables.

Dempster-Schafer Theory

Like Bayesian networks, Dempster-Schafer theory (Gordon and Shortliffe, 1985; Jameson, 1996) is a numerical technique for handling uncertainty. It provides a standard rule for combining pieces of evidence about a single variable which, unlike Bayesian methods, requires no prior beliefs about the associated probabilities. It is most appropriate when the sources of evidence may be unreliable, or when an item of evidence could support a whole set of beliefs. Examples of the use of Dempster-Schafer theory include Carberry's (1990) system for the recognition of users' plans in a natural language consultation, and Bauer's (1996) work on the recognition of email users' plans.

Other Approaches

Other formal approaches to user modelling include machine learning (Michalski, Carbonell and Mitchell, 1983) and fuzzy logic (Zadeh, 1994). Systems employing

machine learning attempt to extract regularities and patterns in a user's behaviour, given a number of observations about the user. These patterns can be used to classify or represent users and their behaviour. For example, Sentance and Pain (1995) applied machine learning techniques to the inference of 'mal-rules' accounting for the article usage errors of learners of English as a second language. Successful applications of machine learning to user modelling have typically been cases where a large number of observations were available (Jameson, 1996).

Fuzzy logic is a technique useful for representing and reasoning with vague or imprecise concepts. It is most often used to mimic human reasoning, or to handle human input which involves such concepts (Jameson, 1996).

3.3 An Adaptive Configuration Support Tool

This section sketches a rough outline of one way in which the fields of adaptive user interfaces and user modelling presented in this chapter could be combined with the assistive technology described in Chapter 2. More specifically, an *adaptive configuration support tool* is proposed. This tool focuses on the software configuration facilities for keyboards and mice, and is intended to support users in finding and setting an appropriate configuration.

In Chapter 2, the most commonly available configuration facilities were described in detail. These are *Sticky Keys*, *Repeat Keys*, *Bounce Keys*, *Slow Keys* and *Mouse Keys*. In addition, many systems allow configuration of the mouse gain and double click speed. Users with physical impairments can find these facilities very useful, but performing configuration can be a demanding process. It requires the user to know what relevant facilities are available, to remember or guess at an appropriate setting, and either to remember the keyboard shortcut which activates each facility or to control the interface well enough using the default configuration to access the facility another way. Novice users are unlikely to know which aspects of the configuration can be altered, and in what ways they would choose to alter them. Occasional users, or those with memory impairments, may have trouble remembering how to activate the facilities. Users who have extreme difficulty with the default configuration are dependent on usable shortcuts, their memories, or assistance from another person.

The problem is compounded when the system being used is shared with others who have different configuration preferences. The user must then perform configuration before every session. Chapter 6 will present evidence that some users in this position perceive the process as too difficult, error prone or time consuming, and choose not to perform configuration at all. Users may also require configuration changes during a session.

Some form of help with configuration would potentially be very useful. Some users do have help, in the human form of a friend, teacher or system administrator. However, this is not the case for all users, and even where such a person is available, they may not be able to choose appropriate configuration settings for the user. For example, it can be difficult for an observer (or even for the user) to tell whether a double letter has appeared because the user pressed the key for too long, or pressed it twice in quick succession.

Automated configuration support could potentially identify which facilities were relevant to a current user, and what settings for those facilities might be appropriate. It could alert novice users to the existence of relevant facilities, explain how to operate them, and take over some of the configuration tasks. This would increase user independence, freeing human helpers to support the actual task for which the computer was being used in the first place, and compensating for situations in which no outside help was available. If users had an easy and reliable way of altering the configuration, then the problems associated with shared machines would also be relieved.

This domain has a number of the features which have been cited (Kass and Finin, 1989; Kühme and Schneider-Hufschmidt, 1993; Rich, 1983) as characterising an appropriate application of adaptive systems and user modelling techniques: a very diverse set of users must be accommodated, the adaptation requirements of individual users may change, and we require the computer system to assume greater responsibility for successful system-user communication.

A number of approaches to building a model of input device configuration requirements could be used. These are:

- *Explicit user questioning.* Users could be supported in choosing an appropriate configuration by a system which questioned them about their difficulties and allowed them to try different settings. This approach would be most effective for users with some familiarity with computers and an awareness of their own

requirements. The mechanism by which users responded to questions would need to be carefully designed so that users having difficulty with the default configuration could still use it. For example, requiring users to point and click with the mouse would be inappropriate for many users. The initial configuration used by the system would need to be as easily accessible as possible. Unfortunately, some configuration facilities can actually hinder users for whom they are not appropriate. It is not clear whether a generally acceptable initial configuration exists.

There are other potential difficulties with this approach. It relies on the user being able to understand their own difficulties and communicate these to the system. It may be difficult for users to articulate their competencies for such low-level skills. Novice or cognitively impaired users may find question sessions particularly difficult, due to limitations in their understanding of the keyboard and mouse, or their language skills. These are often the users who find it difficult to perform their own configuration, and therefore require the greatest support. Question answering is also time consuming for users, and can be off-putting (Rich, 1983).

- *Testing of users on specific tasks.* One way of overcoming difficulties with question asking may be to ask users to perform specific keyboard and mouse tasks, and examine their input for indications of specific difficulties. Task knowledge would make it relatively easy to identify input errors. The major limitation of this approach is that, like question asking approaches, it requires users to invest time and effort in an explicit configuration activity. Some users may be reluctant to invest the time required, while others may be unaware that the activity is available.

- *Modelling users performing their own tasks.* The input provided by users in the course of their computer use could be examined for evidence of physical difficulty, and a model built using this information. The resulting model could trigger a configuration interaction targeted at the particular facilities that have been identified as relevant, or it could be used by the system to alter the current configuration to better suit the user. While error recognition in this scenario is more difficult than when users perform specific tasks, it is attractive in that it represents an opportunity to offer configuration support to users who are unaware that configuration is possible.

In Chapter 2 it was observed that configuration support is often limited. A significant number of users remain unaware of options which could improve the match between their needs and the behaviour of the input devices they use. This observation is supported by the empirical data presented in Chapter 6, Section 6.3. Among the participants who regularly used computers, 50% of those who may have benefited from using *Repeat Keys* and 30% of those who may have benefited from *Sticky Keys* were unaware of these facilities. The third option presented above - that of monitoring users' input unobtrusively with little task knowledge - offers a mechanism by which this gap between the available facilities and their users might be overcome. This thesis therefore investigates the recognition of users' difficulties through monitoring unspecified input tasks. This approach could ultimately be combined with the alternative approaches described above, acting as a trigger for an explicit configuration routine which employed user questioning and further testing. The thesis will show that such an approach is feasible for several important aspects of keyboard configuration. Ultimately, the resulting model could form the basis of a very general support system, suitable for use in many different styles of configuration support.

The proposed adaptive system, using a model of this kind, would implement adaptations to the current configuration of the input devices, and also adapt the configuration suggestions it makes according to the perceived requirements of the current user. The ideal system would initiate adaptation, make suggestions for specific adaptations, and execute the desired adaptations. Responsibility for deciding which adaptations should be implemented could rest with the system or the user. In the classification of Dieterich et al. (1993), this would be a self-adaptive system or a user-controlled self-adaptive system. A combination of the two approaches may be ideal. For example, a system could use self-adaptation to implement changes to the key repeat delay and user-controlled self-adaptation to control the activation of *Sticky Keys*. In the case of the repeat delay, unless key repeats are switched off altogether, changes do not affect the way in which the user interacts with the system, and it may not be necessary for the user to know the current delay setting. This approach has the advantage that the user is not interrupted, while the configuration dynamically alters to suit their key press lengths.

The negative research results reported in Section 3.1.2 suggest that self-adaptation might be inappropriate for facilities which require the user to change the way in which they use the input device. However, it is important to remember that these negative results all derive from studies in which the users had become competent at using a

particular system, and were then required to change. In this application the users are not, and may never become, competent at using the system in its original configuration. Adaptations for the purpose of providing access are perhaps fundamentally different from adaptations designed to improve efficiency, from the user's perspective. It is certainly not clear that changes will be as disorientating as they can be for those with accurate, consistent motor control.

A user-controlled self-adaptive approach has the advantage that the user knows exactly what adaptations have been implemented, and when, and is in control of the operation. Disadvantages are that the user must be interrupted in order to ask whether to implement a given option, and must also understand the option in order to make a decision about it. Empirical experimentation would be necessary to establish the best approach for each type of configuration option. It is possible that additional factors such as individual preferences, the user's task, and the environment of computer use should also play a role in the choice of adaptation strategy.

The preceding discussion has assumed an adaptive system capable of unintrusively and accurately modelling a user's configuration requirements. While individual differences in cognitive approaches to problems are very difficult to infer from a user's input (Thimbleby, 1990), physical input characteristics are, perhaps, more easily measurable, and so the problem of detecting when an adaptation may be appropriate is easier. Within this domain it is not unreasonable to hope to build an accurate model of user characteristics and requirements.

In terms of the user model dimensions identified in the preceding sections, the model should be:

- *Individual* - to meet the very different needs of users.
- *Implicitly constructed* - to accommodate novice users, and others unable to specify their own requirements. In an ideal scenario, users should not be required to perform any special tasks in order to receive configuration support.
- *Dynamic* - because a user's typing behaviour and configuration requirements may change over a single session - users may adopt a different typing style (e.g. begin to use one hand more than the other), become tired, or effect some change in their environment (e.g. altering the height of the table). There may also be different

users within a session: the model should ideally be capable of operating on multi-user machines where there may be no explicit indication of changes of user.

- *Short- or long-term* - Long-term information such as a user's previously preferred configuration, or the variability in their requirements, may be useful. However, in order to accommodate changes in the user's requirements, stored preferences must be compared to some more recent independent assessment. This requires a model with the ability to extract requirements without referring to long-term data, or data gathered from some initial questioning session. Initially, therefore, a short-term model is required. This avoids the difficulties associated with storage and recall of models, particularly in environments where users do not normally identify themselves to the system. This short-term model could then be extended with longer term information as and when necessary.

Chapter 7 will describe a user model with these characteristics, and will discuss the relevance of the user modelling techniques described in the previous section to this problem.

3.4 Summary

Adaptive interfaces offer a mechanism by which users with motor disabilities could be supported in finding and setting appropriate configurations for their input devices. Such support would be particularly useful in increasing user awareness of potentially useful facilities, simplifying the configuration process itself, and tracking changes in user requirements. However, adaptations must be carefully controlled so as not to confuse or alienate users. Self-adaptive systems, which remove all responsibility (and therefore control) from users, have sometimes hindered rather than helped those users. Self-adaptation is nevertheless attractive, since it frees users from the requirement to learn or understand the adaptive process. The Retail User Assistant (Meyer, 1994) is an example of a successful self-adaptive system. It is not clear that the reasons for which users have rejected previous self-adaptive systems would apply in the proposed domain.

Where adaptations necessitate changes in the user's interaction with the system, a greater degree of user control is preferable. Given that users may often be unaware that adaptation is even possible, user-controlled self-adaptation is a promising

approach. A combination of approaches could be used, allowing each different configuration option to be supported in the most appropriate way.

Whatever adaptation style is adopted, adaptation of configuration requires a system capable of making sensible configuration suggestions. These suggestions would be driven by a user model. A number of dimensions along which user models can be classified have been presented. This thesis describes an individual, implicitly acquired, dynamic, short-term model, which assesses a user's requirements for a subset of the existing configuration facilities. A number of successful user modelling techniques are available, including bug libraries, stereotypes and Bayesian networks. These techniques, and their relevance to this application, will be discussed further in Chapter 7. First, however, the domain of input device manipulation difficulties and the corresponding configuration facilities will be explored more fully.

Chapter 4

An Empirical Study of Performance Errors

Given the range of keyboard and mouse access facilities described in Chapter 2, there is surprisingly little published data detailing the actual difficulties that people encounter with keyboards and mice. Chapter 2 has discussed the data which do exist, and the reasons for the current lack of data. This chapter presents an empirical study of keyboard and mouse usage by people with motor disabilities, and outlines the difficulties observed in the group studied.

4.1 Goals of the Study

A detailed investigation of keyboard and mouse usage by people with motor disabilities would provide valuable information on the ways in which difficulties in using these devices manifest themselves in the input stream. In the absence of detailed descriptions of input device difficulties in the literature, an empirical study targeting people with motor disabilities was undertaken. The goal of this study was to identify and examine the performance errors occurring in the use of the standard keyboard and mouse. While there are many alternative input devices that could be considered for this class of users, the standard devices are often preferred where they are usable, for reasons given in Chapter 2.

Information about event timings, keystroke lengths and mouse movements is necessary for the development of automatic procedures for recognising difficulties, or for prescribing appropriate configurations. It may also suggest refinements of, or extensions to, the existing set of configuration facilities.

4.2 Study Methodology

4.2.1 Participants

A total of twenty-six participants (twelve female and fourteen male) aged 25 to 72 took part. Six of these had no physical disability (age range 25 to 35), and provided data for comparison with the participants with disabilities. This group of six is referred to here as the comparison group, while those with motor disabilities form the main group. All participants were unpaid volunteers. Eleven, including all of the comparison group, were recruited through personal contacts; eight responded to an advertisement seeking people with motor disabilities affecting their hands or arms, and the remaining seven were contacted through organisations providing computing for people with disabilities. The first twenty available participants with disabilities were used. The comparison group were selected to represent a range of levels of computer experience. Nine of the twenty-six participants had little or no computer or word processing experience, nine had a moderate or good level of experience, and eight were very experienced. Experience was measured by the participants, who placed themselves on a scale from 'beginner' to 'expert'. Their responses are presented here on the experience scale 'none', 'a little', 'moderate', 'good', and 'expert'. None of the participants were touch typists.

- Participant 1 was a sixty-two year old man whose disability stemmed from a major stroke. He had no use of his left arm and hand, tended to find visual focusing and aim difficult with the keyboard, and found the mouse too sensitive due to tremor in his right hand. He had moderate computer and word processing experience, gained before and after his stroke. His computer usage consisted of a weekly session as a volunteer worker for a charity. There he used a Macintosh with *Sticky Keys* activated, the key repeat deactivated, and a standard Apple keyboard and mouse to edit a database.

- Participant 2 was a sixty year old man with incomplete tetraplegia, who had difficulty flexing his fingers. He usually used a PC running Windows 95 at home for word processing, about once a week, and had four years of computer experience. He typed using a prodder held in his right hand, and usually used *Sticky Keys*, but could also use his left hand with some effort. He reported finding the standard PC mouse difficult, particularly for dragging.

- Participant 3 was a forty-five year old man with neurological damage as a result of surgery on a spinal cord tumour. This caused increasing muscle wastage and spasticity. He had been using computers for 28 years and used a Macintosh at home, with a mouse with a 'drag' button, and a PC at work. He also used *Mouse Keys* and a trackerball as mouse alternatives. He typed with several fingers of both hands, and wore a finger splint on the first finger of his right hand.

- Participant 4 was a forty-eight year old man with incomplete tetraplegia as a result of an early rugby accident, which impaired dexterity in his hands. He had been using computers for 29 years. He usually used a PC, but also used Macintosh and UNIX platforms. He typed predominantly with his right hand, but used his left for control keys.

- Participant 5 was a seventy-two year old woman with proprioceptive difficulties due to peripheral nervous system damage as a result of chemotherapy and exposure to radiation. She had been using Macintosh computers for one year, in a class targeted at people with disabilities. She typed with one finger of her left hand only, and usually used *Sticky Keys*.

- Participant 6 was a thirty-eight year old man whose left hand was affected by radial palsy and tendon transfers. His right hand was unaffected. He had never used computers before. To demonstrate the difficulties he experienced with his left hand, he typed and used the mouse with this hand, using his right hand only for modifier keys.

- Participant 7 was a retired woman who had loss of feeling and movement in her wrists, which had been badly broken in the past. She had never used a computer before. She typed with both hands.

- Participant 8 was a retired man who had muscle wastage affecting both hands in a way similar to arthritis. He had never used a computer before. He typed with his right hand only, predominantly with the first finger.

- Participant 9 was a woman in her thirties who had recently had a right shoulder replacement. She had moderate computer experience on a Macintosh. She was normally a touch typist, but due to her operation was restricted to typing mainly with her left hand.

- Participant 10 was a man in his fifties who had suffered a major stroke three years previously. This deprived him of the use of his right hand and arm, and affected his memory, his ability to generate spoken and written language, and his powers of concentration. He had moderate computer experience prior to his stroke. He typed with his left hand only.

- Participant 11 was a woman in her fifties who had had a stroke which affected her right side. She also had myelitis (a form of multiple sclerosis) and experienced double vision and difficulty with co-ordination. She had a little computing experience, gained prior to her stroke. She typed with her left hand only.

- Participant 12 was a thirty-seven year old man with cerebral palsy. He was moderately experienced in the use of computers and usually used PCs at home and for voluntary work with a local charity. He found the mouse awkward and preferred to use a mouse with a button for double click. For his voluntary work, he used a standard PC mouse. He typed predominantly with his right hand, using his left for modifier keys.

- Participant 13 was a forty-nine year old man. He experienced spasms and difficulty in co-ordination, caused by having been prescribed the wrong medication for epilepsy. He was moderately experienced in computer use, and was undertaking a word processing course, using PCs running Windows 3.11. He typed using both hands.

- Participant 14 was a sixty-seven year old woman with pronounced tremor in both hands. She had a little computer experience, based on PCs running Windows 3.11. She typed mainly with her right hand, which was less shaky than the left.

- Participant 15 was a forty-year old man with cerebral palsy, with 34 years of computer experience using PCs and Sun workstations. He typed with his left hand only, usually used a standard mouse, but sometimes used *Mouse Keys*.

- Participant 16 was a sixty year old man with polymiocitis, which caused general muscle loss and weakness. He was an expert Macintosh user, and usually used a standard mouse or a trackerball, depending on his current task. He typed with both hands.

- Participant 17 was a twenty-seven year old woman with spina bifida, causing impaired right hand function and right spastic hemiplegia. She had a little computer experience, gained through using BBC Microcomputers[1] and an Apple Macintosh at her local day care centre. She typed using her left hand only.

- Participant 18 was a man of unspecified age and disability. He had a little computer experience, based on the BBC Microcomputer at his local day care centre. He had never used a mouse before. He typed using both hands.

- Participant 19 was a thirty-three year old woman with cerebral palsy and RSI. She had 28 years of computer experience on Macintosh and PC platforms, and most often used a Macintosh. She sometimes used *Sticky Keys* on a portable computer, and sometimes used *Mouse Keys* on her Macintosh. She also used a slowed key repeat delay, and mouse tracking speed, and had her keyboard sloped at a comfortable angle.

- Participant 20 was a twenty-nine year old woman with cerebral palsy. She was an experienced PC user. She usually used a keyguard, *Sticky Keys*, and *Mouse Keys*, and rarely used the mouse itself. She typed with both hands.

- Participant C1 was a twenty-seven year old woman with no physical disability. She had never used a computer before.

- Participant C2 was a thirty-five year old man with no physical disability. He was very experienced with PCs but had never used a Macintosh before.

- Participant C3 was a thirty-one year old man with no physical disability. He was very experienced in the use of PCs.

- Participant C4 was a thirty-one year old woman with no physical disability. She was moderately experienced in using PCs.

- Participant C5 was a twenty-six year old woman with no physical disability. She had a little computer experience, based on a PC platform.

---

[1] The BBC microcomputer is a pre-IBM PC popular in Britain in the 1980's. No mouse is used, input being solely via the keyboard.

Table 4.1: The participant sample

Participant	Experience	Disability	Typing Technique	Mouse Hand
1	moderate	stroke	right hand only	right
2	good	incomplete tetraplegia	mainly right hand, prodder	right
3	expert	spasticity, weakness	both hands, several fingers	left
4	expert	incomplete tetraplegia	mainly right hand	right
5	moderate	proprioceptive disorder	left hand, mainly 1st finger	left
6	none	radial palsy	mainly left hand	left
7	none	wrist stiffness	both hands, several fingers	right
8	none	muscle wastage	right hand, mainly 1st finger	right
9	moderate	shoulder replacement	both hands, several fingers	left
10	moderate	stroke	left hand only	left
11	a little	stroke/myelitis	left hand, mainly 1st finger	left
12	moderate	cerebral palsy	mainly right hand, several fingers	right
13	moderate	co-ordination loss, spasms	both hands, 1st two fingers	right
14	a little	shakiness	mainly right hand	right
15	expert	cerebral palsy	left hand only, several fingers	left
16	expert	muscle loss, weakness	both hands, middle fingers and thumbs	left
17	a little	spina bifida	left hand, 1st two fingers and thumb	left
18	a little	muscular weakness	both hands, any digit	right
19	expert	cerebral palsy, RSI	both hands, any digit	right
20	expert	cerebral palsy	both hands, any digit	right
C1	none	none	both hands, 1st two fingers	right
C2	expert	none	both hands, any digit	right
C3	expert	none	both hands, several fingers	right
C4	moderate	none	both hands, any digit	right
C5	a little	none	both hands, any digit	right
C6	moderate	none	both hands, any digit	left

- Participant C6 was a twenty-five year old woman with no physical disability. She was a novice computer user, whose prior experience was based on PCs.

Table 4.1 summarises each participant's experience, disability, typing technique and the hand they used to operate the mouse. The participants with disabilities were recruited through a variety of sources, none of which were focused on a specific type of disability, in order to provide as representative a sample as possible. However, because of the very large range of difficulties due to motor disabilities, a small group such as this cannot be truly representative. Instead, it can point to specific problems which do occur, providing a preliminary indication of their impact and importance.

The comparison group are all university graduates (C1-C5) or students (C6), and are all in their twenties or thirties. They are not intended to and do not form a representative sample of non-disabled computer users, and their results may not apply to, for example, users of different age groups, or to touch typists. They are included simply as a reference point for comparison with the participants with disabilities.

4.2.2 Materials and Apparatus

Three computer input tasks were used. The first was a typing task, consisting of a text passage to be copied. The passage was presented as double spaced, typed text. It was approximately 100 words in length, and required a minimum of 547 key presses, including 28 uses of the *Shift* key, eleven of which were upper case punctuation marks for which the *Shift* key could not be used. The passage, shown in Figure 4.1, deliberately included characters on all parts of the keyboard (excluding the numeric key pad and function keys), and required the use of the *Shift* key in conjunction with keys in a variety of positions. It did not contain repeated characters inviting the use of the key repeat facility, although repeats would naturally be used if making corrections with the arrow keys.

The second task was entirely mouse based. It consisted of 17 pointing, clicking, multiple clicking and dragging operations. These were presented as a list of tasks and were performed on a text passage in which all but the target words were obscured.

The targets of the operations were menus and words in fixed positions on the screen. They varied in size from 3 pixels wide to the whole width of the text window, and involved several different targets distributed over the majority of the screen.

There was a British grandfather called Quentin who said that he was 101 years old. "Are you sure?" asked his friend Maxine. She was 16.

"Yes! I was born in 1895" he replied. But Maxine added in her head the sum 1895 + 101 = 1996, and knew that Quentin was actually 100 this year. He is younger than he thinks (but not by much). Perhaps he has forgotten what year it is. I do that sometimes too. Do you?

You may worry about forgetting what the current year is. Zinc in the diet is supposed to help the memory, but who knows!

Figure 4.1: The typing task

The third task was an editing task, and involved a combination of keyboard and mouse usage. It consisted of 27 typing and mouse operations, including the use of scroll bars, selection from hierarchical menus, use of a dialogue box, and use of the *Command* key. The tasks were based on a text passage containing errors to be corrected. Again the targets were in known positions on the screen.

A session record form was designed and used to document information about each participant, including their previous experience, disability, preferred input devices, ease of performance of the tasks, and level of fatigue experienced. All three tasks, the passages on which they are based, and the session record form are reproduced in Appendix A.1.

All participants used the ClarisWorks® word processor, version 4. For the purposes of recording keystrokes and mouse movements, custom logging software - *InputLogger* (Trewin, 1998) - was developed. This software was Macintosh specific, since Windows 95 was not yet available at the time it was developed, and Windows

3.11 would not support the low level event access necessary. The format of the log files produced by *InputLogger*, and an example log file fragment, are provided in Appendix B.1.

In order to record errors in as realistic a setting as possible, the participants' own computer and venue were used wherever possible. The *InputLogger* software was Macintosh-specific, so this was only achieved when the participant usually used a Macintosh machine at a venue at which a recording could be made. Participants 5, 16, 17, 18 and 19 used machines and/or venues with which they were familiar. Participant 5 used a Macintosh 475 8/150 machine with a domestic Apple keyboard and a curved single button mouse in which the button occupied the whole of the top part of the upper mouse surface. Participants 16, 17 and 18 used a Macintosh with an Apple ISO extended keyboard, in a computer room at their local day care centre. Of the three, only Participants 16 and 17 had used that particular machine before. Participant 19 used a Power Macintosh with an Apple ISO extended keyboard positioned on a slope, in a private office at her workplace. The mouse used was a flat mouse with a rectangular indented button. The remaining participants used the same venue and platform as that described for Participant 5, which was located in a facility designed to allow disabled access. Additional equipment such as adjustable height tables and wrist rests were available there, and were used to adapt the environment to suit each individual as far as was possible.

The use of alternative venues means that the results are less tied to the specific context of the equipment used at the main venue, while still remaining Macintosh specific. Since the aim of this study is to establish as broad a picture as possible of performance errors that do occur, a wider range of venues and platforms would have been more satisfactory. It is possible that some of the difficulties observed were tied to the particular keyboard or mouse design used, and that alternative designs would encourage errors not observed in this study. Unfortunately, a multi-platform study was not practical. These results, therefore, will apply best to users of Macintosh keyboards and mice.

Because many participants were used to alternative platforms, the results here will also be affected by negative transfer of learning effects. This difficulty is discussed further in Section 4.3.3.

The majority of participants did not know their ideal mouse gain setting or double click speed. For them, the mouse tracking was initially set to sensitive, and the mouse

double click speed was on the median setting. The sensitive gain setting was used with the expectation that this would amplify difficulties in pointing. An exception was Participant 19, who used a lower mouse tracking speed - her usual setting. The default keyboard configuration was used - *Sticky Keys* and *Slow Keys* were inactive - with the exception that the key repeat delay was varied between participants (see Section 4.5.1).

A video camera mounted on a tripod was used to record each task.

4.2.3 Procedure

Each session was administered in the same way by the same experimenter, regardless of the venue used. First, each participant was made as comfortable as possible. This involved adjusting the table height, providing wrist rests, positioning the keyboard appropriately, and preventing it from moving around. In some cases further adaptations were made as the experiment progressed. The only external aid that was not permitted was the keyguard. Keyguards prevent the majority of performance errors, while this investigation focuses on those same performance errors for users typing at their natural speed.

The first task was displayed next to the screen, and the readability of both the task text and the text on the screen were checked. For all participants, 18 point text was used. The video camera was focused on the keyboard. The terms to be used in the experiment, such as 'point', 'double click', 'scroll bar' and 'menu' were explained to those participants for whom they were unfamiliar. Novice participants were also given a demonstration of clicking, dragging and other relevant operations. Participants were then allowed a short practice session, with further instruction where necessary. The effect of unfamiliarity with computers, or of familiarity with alternative systems, is discussed in Section 4.3.3.

Participants were then told that there were up to five tasks to perform, and that the number of tasks they actually did would depend on time, and how they were feeling. They were told that they would be asked to perform a keyboard task, a mouse task and an editing task, and then they would repeat the mouse task and the keyboard task.

Because this study was concerned with physical input errors, it was desirable to minimise other errors such as misunderstanding the task. Ideally, this would have been achieved by allowing participants to practice the tasks prior to recording. However, because the experimental sessions were limited to two hours, and many participants became fatigued in less than two hours, providing long practice sessions would have greatly reduced the volume of data recorded, and the beneficial effects of practice may have been counterbalanced by detrimental effects of cognitive fatigue. In addition, the goal of the study was to examine as wide a range of keyboard and mouse operations as possible. For this reason, tasks which repeated the same operation many times were also not appropriate. As a compromise, participants were asked to perform the same set of keyboard and mouse tasks twice.

Participants were told to perform the tasks in their own time, and that they were free to rest or stop as they chose. When ready, video recording was initiated and the session began. First, participants were asked about their previous computer and word processing experience, and the input devices and configuration they normally used.

The typing task was then presented at the side of the screen, mounted on a vertical document holder. Participants were asked to copy the passage as accurately as possible, without performing any error correction. The emphasis on accuracy was intended to minimise errors due to carelessness, and to encourage participants to try to perform operations they may normally find difficult and avoid, such as the use of modifier keys for upper case punctuation. The intention was to record an example of the 'best performance' for each person.

Input logging was initiated, and the experimenter was positioned at the side of the participant, observing their typing and noting any errors that occurred. The experimenter also provided help and advice, should the participant require assistance in understanding the task or the use of the keyboard. When the task was completed, the experimenter stopped input logging.

The experimenter then opened the appropriate task file, and the mouse task was presented as a list of instructions placed by the side of the screen. The video camera was focused onto the screen and input logging was initiated. As the participant performed the tasks, the experimenter provided any necessary assistance with task comprehension, and recorded any difficulties occurring. If necessary, the mouse tracking speed was reduced (one participant), or the participant switched to using *Mouse Keys* (two participants). Upon completion of the task, logging was stopped.

The experimenter then presented a different text passage on the screen, and displayed the editing task instructions. Input logging was initiated. Again, instruction was provided where necessary, and the experimenter noted difficulties experienced by the participants. The editing task provides a different, more realistic context for performing the operations examined in the typing and mouse tasks.

The mouse task was then re-presented, logged and observed. Finally, the video camera was focused on the keyboard, and the typing task was re-presented, logged and observed, this time with error correction allowed. Participants were asked to ignore or correct errors in the way they normally would. No automatic spelling corrector was available. If novice participants chose to correct errors and were unsure how to go about it, instruction was given. This task provided examples of difficulties and confusion occurring during error correction, and is a more realistic sample of typing.

Participants were then asked to rate their ease of performance of a number of individual tasks on a seven point scale with values ranging from 'easy' to 'impossible', and were asked whether they had experienced fatigue during performance of the tasks. Video recording was then stopped.

Sessions lasted for up to two hours, extended only if the participant chose to continue. Consequently, 10 participants did not complete all of the tasks.

4.2.4 Data

The following data was recorded for each participant:

- *An automatically generated log of input event*s for each task, produced by *InputLogger* (Trewin, 1998). This consisted of time-stamped input events, including key up and mouse movement events. Mouse events were associated with a screen position and time. Event timings are recorded to the nearest 1/60 second. Mouse movement while dragging cannot be recorded by *InputLogger*.

- *A video of the participant performing the tasks.* This was used to help in establishing the actual performance errors that occurred. For the typing tasks, the video was focused on the keyboard. For the mouse and editing tasks, the video was focused on the screen.

- *Observations made during the word processing tasks.* For each participant, the same observer recorded impressions and particular examples of the keyboard and mouse difficulties experienced by the participant.

- *Background information about the participant.* This included the nature of their disability, their previous experience with computers and word processors, the set of configuration options they usually used (if known), and their self-reported levels of fatigue and ease of performance of the tasks.

4.3 Analysis

Analysis covered two aspects of computer usage: the ease with which the devices can be used, measured through participant reports; and the errors occurring through difficulties in using the devices, which are examined in terms of frequency of occurrence and the timing information associated with them. A third aspect of the data worthy of study is the time taken to activate a key. This has not been examined in detail, although an impression can be gained from the timing information for the typing tasks reported in Table 4.2 (page 79) and the times taken to perform the mouse tasks, reported in Table 4.7 (page 92). While this timing information is interesting, it includes many factors not related to physical disability, including the time taken to read and understand the next task; the time to find letters on the keyboard; time to transfer attention between the task, the keyboard or mouse, and the screen; time spent checking for errors; and time pausing to ask questions or to clarify task instructions. The effects of these will vary considerably among the group studied, depending on factors such as previous experience and disability. The methodology employed did not allow these effects to be measured, and so no detailed analysis of this timing information has been attempted.

Errors which occurred were manually annotated in the log files. These annotations indicated the type of error, differentiating between classes of performance error, and placing all other errors into a single class. This class included errors caused by misunderstanding the task, or forgetting to perform some operation (e.g. releasing the *Caps Lock* key after typing a capital letter), and spelling errors. They were identified by the participants' own comments, in addition to the video evidence, as described

more fully below. The set of annotations, their format, and an example annotated log file fragment can be found in Appendix B.2.

The annotated log files were then automatically filtered and the Systat® statistical package (SYSTAT, 1992) used to perform the analyses. Non-parametric statistics were used, as the variables under examination do not have, or cannot be assumed to have, normal distributions.

4.3.1 Analysis of Keyboard Errors

Keyboard errors were primarily identified by keystroke events which differed from the input dictated by the task. A keyboard error occurred wherever an intended key was missed, an unintended key was hit, a key was pressed for the wrong period of time, or keys were pressed in the wrong order. The observations and video evidence were used to classify these errors according to their underlying cause, and to extend the set of errors to include those not detectable from the log file, such as cases where a participant's attempt to press a key failed, producing no input. The resulting classes of keyboard performance error are described in Section 4.5.

For missing characters, the video was used to identify whether the participant had made an attempt to press the key or had not attempted to press the key at all. For additional characters, the video was again used to identify whether the key was pressed with the same digit or body part as a deliberate key press attempt; the key was accidentally pressed, perhaps as a result of a spasm or leaning on the keyboard; or the character was generated by a deliberate key press.

4.3.2 Analysis of Mouse Errors

When using the mouse, the boundary between correct performance and an error is often dependent on the context in which the action is being performed. For example, when dragging the box in the scroll bar, the cursor position when the mouse button is raised may be a little distance away from the scroll bar itself, and the operation will still be successful. When dragging to select a piece of text in the middle of the screen, on the other hand, the positioning must be very accurate.

The basic mouse operations are pointing, clicking and dragging. Pointing and dragging have been analysed in terms of the time taken to perform the operations, and the accuracy with which the target was pinpointed. Timings are most likely to be useful when dragging, since it is unlikely that the user will pause in the middle of a dragging task.

Accuracy of mouse movements has been measured in terms of the distance of the relevant mouse event from the nearest position on the screen that would have resulted in a successful operation. In the mouse and editing tasks, and in word processing in general, targets can vary greatly in size. When dragging, for some targets it is acceptable to raise the mouse some distance outside the target area shown on the screen: the scroll box is one common example. Although the mouse up position appears to be outside the target, the result is not considered an error, since visual feedback indicates to the user that their action will have the desired effect. Such targets are referred to in this thesis as *loose*.

Analysis of clicking and multiple clicking is based on the observation that the mouse should ideally remain still while clicking. This analysis examines the movement within all mouse clicks, and between the clicks of multiple clicks, regardless of whether or not an error actually occurred. The range of mouse movement acceptable within a click is dependent on the target size and starting position within the target. Even one pixel of movement may cause an error.

4.3.3 Previous Experience and Practice Effects

Nine of the participants, including two members of the comparison group, had little or no computer or word processing experience. For these participants, the practice session provided was vital. After this short session, five of these participants showed no observable problems due to inexperience. For four others (Participants 6, 8, 11 and 18) the practice session was not enough, and some difficulties attributable to lack of experience were observed, particularly in mouse dragging tasks. For example, participants sometimes abandoned a drag operation because they did not understand how to complete it. It was often clear from their own comments and questions when this had happened. If the observer identified an error of unclear cause, the participant was prompted for a description of the reason for performing the operation in that way. Errors due to inexperience have not been classified as performance errors.

Some wrong inputs were also caused by negative transfers of learning from previous word processing experience. Ten participants (including three of the non-disabled participants) had previous experience that may have conflicted with the operational requirements of ClarisWorks. For example, the version of ClarisWorks used in the experiment displayed menus only while the mouse button was held down. For participants who were used to word processors where the menus remained open after clicking on them, a click on a menu item was, in the absence of evidence to the contrary, interpreted as a deliberate click rather than a failed attempt to drag.

Seven participants were used to using differently designed mice, or configuration options such as *Sticky Keys*. These differences also caused negative transfer errors, even though practice was given. Where identifiable, such errors have not been classified as performance errors.

During the course of the study, practice effects were also observed, as participants became familiar with the keyboard, and the tasks. This study does not examine the effect of practice on the errors observed. The goal is to investigate difficulties that occur for both novices and more experienced users, and all of the errors observed are equally important.

4.4 Pilot study

The first four participants, all of whom had some motor disability, formed a pilot test for the remainder. After the first two participants had performed the tasks, the mouse and editing task instructions were simplified, and the base passages edited to allow for easier identification of targets. It was also found that performing operations at the edge of the text window, such as menu access, encouraged errors which caused the text window to move. Because data analysis depended on prior knowledge of the target positions on the screen, this made analysis difficult. Menu access and scrolling operations were therefore restricted to the editing task, to facilitate easier analysis of the mouse tasks.

At this stage, the task ordering was also finalised. Initially, the two typing tasks had been presented first, followed by the mouse and editing tasks. This ordering minimised the need to refocus the video camera. However, it appeared that placing similar tasks consecutively exacerbated problems with fatigue. The final task order

was therefore chosen to separate similar tasks, while requiring the video camera to be focused only twice.

Participants 3 and 4 used the revised methodology, and it was carried forward unchanged into the main study. The typing tasks were not changed, therefore typing data for all 26 participants was available. For the mouse and editing tasks, the first two participants were excluded from analyses of pointing, where target knowledge was required, but included in click, multiple click and drag analyses.

Preliminary results from the pilot study have been published in Trewin (1996).

4.5 Keyboard Results

Nineteen participants attempted both typing tests and seven only one of the tests. All tests were completed except for the second typing test for Participant 11, who stopped due to fatigue. One typing log for Participant 2 was lost through experimental error. A total of 43 complete log files and one incomplete log file were available for analysis. In addition, some typing data from the editing tasks was also available for twenty-four participants. Four participants did not complete all of the typing in the editing tasks, due to fatigue and time limitations, while Participants 1 and 2 performed a different version of the editing task, which contained very little typing. The volume of typing data recorded in the editing tasks varied between participants. For simplicity, and in order to allow between participant comparisons, the editing task data has been excluded from Tables 4.2 and 4.3, but it has been included in calculations not related to a specific task, such as the overall mean and standard deviation of each participant's key press lengths.

Table 4.2 summarises the two typing tests, labelled T1 and T2, and includes data for the main (Participants 1-20) and comparison (Participants C1-C6) groups. For each participant, the total number of keystrokes made and the number of minutes taken to complete each task are given. In the main group, the times varied from 53.2 minutes for Participant 13's second task to 3.8 minutes for Participant 3's first task, the average being 14.0 minutes for the first task and 16.8 minutes for the second. For the main group, there was an inverse relationship (Spearman Rho = -0.765, $p<0.05$) between the times taken and experience level for the first task. When error correction

Table 4.2: Summary of the typing tasks

Participant	Keystrokes recorded		Time (minutes)		Performance errors (% time)		Other errors (% time)	
	T1	T2	T1	T2	T1	T2	T1	T2
1	588	1348	15.5	30.7	0.0	18.1	0.0	14.1
2	558	.	6.5	.	0.0	.	0.0	.
3	550	637	3.8	5.1	0.0	18.2	0.0	5.4
4	565	559	4.6	4.5	0.0	0.0	0.0	0.0
5	547	567	8.0	7.7	0.0	3.8	0.0	4.8
6	556	590	27.5	24.0	0.0	3.0	0.0	0.7
7	561	594	17.5	14.9	2.0	16.2	0.0	6.3
8	660	.	36.6	.	0.0	.	0.0	.
9	572	608	6.7	7.6	0.9	8.8	0.7	5.3
10	.	782	.	49.3	.	8.9	.	5.3
11	647	325	8.2	4.4	0.0	0.0	0.0	0.0
12	593	556	16.2	13.2	1.7	2.1	1.9	1.4
13	.	766	.	53.2	.	10.8	.	8.7
14	552	.	17.1	.	0.0	.	3.3	.
15	605	575	14.8	14.7	4.4	0.6	1.2	0.1
16	582	583	5.7	5.5	0.0	0.6	0.0	4.8
17	597	.	28.0	.	0.0	.	0.0	.
18	571	.	14.9	.	0.0	.	0.0	.
19	574	620	5.2	5.9	0.0	9.8	0.0	2.2
20	624	626	16.6	11.3	3.4	8.2	3.6	2.7
Average:	585	649	14.0	16.8	0.7	7.3	0.6	4.1
C1	.	551	.	11.8	.	0.0	.	8.0
C2	557	566	3.0	3.2	0.0	0.6	0.0	0.0
C3	562	568	3.8	3.8	0.0	2.0	0.0	4.2
C4	554	566	3.9	4.2	0.0	1.5	0.0	15.8
C5	537	593	9.7	8.8	0.0	4.5	0.0	3.8
C6	555	569	3.9	4.4	0.0	0.0	0.0	3.6
Average:	566	569	4.9	6.0	0.0	1.7	0.0	5.9

was introduced in the second task, the relationship was weaker (Spearman Rho = -0.407, not statistically significant).

Among the comparison group the average time was 4.9 minutes for the first task and 6.0 minutes for the second task, with a strong inverse relationship between experience level and time taken in both the first (Spearman Rho = -0.985, $p<0.01$) and second (Spearman Rho = -0.949, $p<0.05$) tasks. The difference in average times for the two tasks is largely due to Participant C1, a novice keyboard user who typed much more slowly than the others, and did not do the first typing task due to lack of time. Typing samples for Participant C1 and a number of other participants are given in Appendix C.1.

Also shown in Table 4.2 is the time spent correcting performance errors and other errors, given as a percentage of the total time taken. The class of 'other errors' includes all of the wrong inputs which are not classed as performance errors, as described in the previous section. Error correction times should be zero for task one for all participants, but some participants did make corrections. Participants 4 and 11 appeared not to notice the few errors they made in the second typing task, and made no corrections. Participants C1 and C5 made no performance errors in their second typing tasks. Where errors were corrected, the average time spent correcting performance errors was greater than the average time spent correcting other errors for the main group, and less for the comparison group.

In the typing tasks for the main group, examples of seven different performance errors were observed. One example of an eighth performance error - that of holding down the *Shift* key for too long - was observed in the comparison group, but is excluded from this discussion. In order of frequency, the errors recorded were:

1. *Long Key Press Errors*: An alphanumeric key was pressed for longer than the default key repeat delay.
2. *Additional Key Errors*: A key near to the intended key was activated by the digit or other part of the body that was intended to activate the desired key. The desired key itself may or may not have been pressed.
3. *Missing Key Errors*: A movement intended to press a key did not produce a character, either because the participant's aim was off target (perhaps causing additional key errors), or because the key was not pressed with sufficient force.
4. *Dropping Errors*: The participant failed to press two keys simultaneously (e.g. use of the *Shift* key).
5. *Bounce Errors*: The participant unintentionally pressed the intended key more than once.
6. *Remote Errors*: The participant, while trying to press a key, accidentally pressed a different key with a digit or body part other than the one being used for the intended key press. Other accidental key presses, such as leaning on a part of the keyboard, are also remote errors.
7. *Transposition Errors*: Two keys were transposed.

Table 4.3: Performance errors in the typing tasks

par-ticipant	no. of shifts	long key presses		additional		missing		dropping		bounce		remote		transposed	
		T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2	T1	T2
1	8	0	0	38	35	11	6	0	3	0	0	0	0	0	0
2	.	1	.	0	.	0	.	0	.	0	.	0	.	0	.
3	56	30	35	9	4	10	3	0	1	0	0	0	1	0	2
4	56	0	0	4	0	2	1	8	6	0	0	2	0	0	0
5	56	13	39	6	4	4	4	0	0	0	0	0	0	0	0
6	22	31	27	2	2	12	18	2	3	8	12	2	0	1	0
7	56	37	23	11	3	8	3	4	0	0	0	10	2	0	0
8	28	114	.	5	.	4	.	0	.	0	.	2	.	0	.
9	56	16	39	13	6	11	5	0	2	0	1	0	0	0	0
10	11	.	377	.	3	.	0	.	0	.	0	.	0	.	0
11	11	1	0	4	1	1	0	0	0	0	0	0	0	0	0
12	56	311	171	2	1	1	1	4	2	2	1	0	1	0	0
13	28	.	510	.	4	.	4	.	3	.	3	.	0	.	0
14	28	14	.	0	.	0	.	0	.	0	.	0	.	0	.
15	22	14	44	8	8	10	8	11	2	3	3	1	7	0	0
16	56	0	0	3	1	4	1	0	0	0	0	0	0	0	0
17	0	45	.	4	.	1	.	0	.	0	.	0	.	0	.
18	11	9	.	0	.	0	.	0	.	0	.	0	.	0	.
19	56	299	357	28	19	7	8	0	0	3	0	2	2	0	0
20	56	36	18	23	14	20	13	2	2	5	3	5	0	0	1
Total		2610		265		179		55		44		37		4	
C1	28	0	.	0	.	0	.	0	.	0	.	0	.	0	.
C2	56	0	0	2	2	1	1	0	0	0	0	0	0	0	0
C3	56	0	0	0	3	0	0	0	0	0	0	0	0	0	0
C4	56	0	0	2	1	1	1	0	0	0	0	0	0	0	0
C5	56	0	0	0	1	1	5	0	1	0	0	0	0	0	1
C6	56	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Total		0		11		10		1		0		0		2	

For all performance errors except long key press errors, Table 4.3 gives the number of errors of each type observed in the typing tasks. The numbers of long key press errors given are not actual errors occurring, but the number of key presses intended to produce a single character that would have repeated under the default key repeat delay (See Section 4.5.1).

Participants were also asked to rate how difficult they found it to press two keys at once (Shift), reach all the keys on the keyboard (Reach), aim accurately at a key (Aim), only press a single key (Isolate) and to press keys quickly (Fast). Answers were on a scale ranging from 'easy', through 'some difficulty', 'moderate difficulty', 'hard', 'very hard', 'extreme difficulty' up to 'impossible'. Table 4.4 gives each participant's answer for each category. Difficulty in pressing two keys at once may cause dropping

Table 4.4: Reported ease of keyboard manipulation

Participant	Shift	Reach	Aim	Isolate	Fast
1	impossible	easy	some diff	moderate	easy
2	easy	easy	easy	easy	easy
3	easy	easy	some diff	some diff	easy
4	moderate	easy	some diff	some diff	easy
5	hard	easy	some diff	some diff	easy
6	impossible	moderate	easy	easy	easy
7	easy	easy	easy	some diff	some diff
8	easy	easy	easy	easy	easy
9	some diff	some diff	some diff	easy	easy
10	very hard	easy	easy	easy	hard
11	impossible	easy	easy	easy	easy
12	some diff	easy	easy	easy	easy
13	easy	easy	easy	easy	very hard
14	easy	easy	easy	some diff	hard
15	hard	moderate	some diff	some diff	hard
16	easy	easy	some diff	easy	easy
17	easy	easy	easy	easy	easy
18	easy	easy	easy	easy	easy
19	moderate	easy	moderate	moderate	extreme
20	moderate	some diff	some diff	easy	easy
C1	easy	easy	easy	easy	easy
C2	easy	easy	easy	easy	easy
C3	easy	easy	easy	easy	easy
C4	easy	easy	easy	easy	easy
C5	easy	easy	easy	easy	easy
C6	easy	easy	easy	easy	easy

errors, or avoidance of the use of the *Shift* key. Difficulty in reaching all parts of the keyboard may result in additional key errors, remote errors, or missing key errors. Difficulty in aiming could cause additional key or missing key errors. Difficulty in isolating a key to press could also be a cause of additional key errors. Difficulty in pressing keys quickly could lead to long key press errors.

4.5.1 Long Key Press Errors

On a Macintosh, the default delay before a key will repeat is 16 ticks (1 tick = 1/60 sec). For many people, this is too short. Unwanted extra copies of a letter are long key press errors.

Because there is so little existing data describing long key press errors, the study was not only required to capture information about the frequency of such errors, but also

about time and effort spent correcting such errors. However, it was predicted that for some participants correction of these errors could potentially occupy the greater portion of the time allocated to the entire experiment. As a compromise, the majority of long key press errors were suppressed by using a long repeat delay, or by disabling the key repeat facility. The key repeat delay in force can be altered without affecting the events recorded by *InputLogger*. For eight of the main group (Participants 1 - 6, 15 and 20), the key repeat facility was disabled, the intention being to discover each participant's 'natural' key press length, in the absence of any requirement to raise keys quickly. However, this did not allow recording of information about the correction of such errors. To provide data concerning long key press error correction, the remaining twelve of the main group performed the typing tasks with a key repeat delay of 24 ticks - longer than the default delay. The comparison group were not expected to make long key press errors, and all performed the tasks with a key repeat delay of 12 ticks.

Because of the differences in repeat delay settings used by the participants, this analysis does not report the actual number of errors that occurred, but the number of errors that would have occurred had each participant typed the same keystrokes using the default key repeat delay of 16 ticks. The actual numbers of errors were zero for Participants 1-6, 15 and 20, since the key repeat facility was disabled. For Participants 7-14 and 16-19, a long repeat delay was in force, so the actual numbers of errors occurring were much smaller than the numbers in the table (the maximum being 43 for Participants 10 and 13). No actual or projected errors were recorded in the comparison group.

Table 4.5 details, for each participant, their reported difficulty in pressing keys quickly, their average key press length, the standard deviation of their key press lengths, the number of key presses longer than the default key repeat delay for each typing task, and the repeat delay they used. In order to examine only key presses intended to produce a single character, these values are measured using alphanumeric and punctuation keys only. The *Delete* key, the arrow keys, and the *Return* key are excluded from the calculation because the repeat facility is often deliberately used on these keys. Modifier keys, remote errors, additional errors and bounce errors are also excluded.

Table 4.5: Key press length summary

Participant	Reported difficulty	Average key press length (ticks)	Standard deviation of key press lengths	Long key presses		Repeat delay used (ticks)
				T1	T2	
1	easy	7	1.66	0	0	off
2	easy	5	1.09	1	.	off
3	easy	9	3.87	30	35	off
4	easy	4	1.03	0	0	off
5	easy	6	5.26	13	39	off
6	easy	10	5.78	31	27	off
7	some diff	11	3.76	37	23	24
8	easy	12	3.92	114	.	24
9	easy	9	3.68	16	39	24
10	hard	17	5.81	.	377	24
11	easy	5	2.68	1	0	24
12	easy	15	2.97	311	171	24
13	very hard	20	5.27	.	510	24
14	hard	10	2.84	14	.	24
15	hard	10	3.24	14	44	off
16	easy	5	1.39	0	0	24
17	easy	10	10.2	45	.	24
18	easy	9	9.88	9	.	24
19	extreme	16	3.61	299	357	24
20	easy	11	3.07	36	18	off
		Ave:10	Ave: 4.05	Total: 2610		
C1	easy	8	1.35	0	.	12
C2	easy	5	1.22	0	0	12
C3	easy	4	0.95	0	0	12
C4	easy	5	1.60	0	0	12
C5	easy	4	0.98	0	0	12
C6	easy	5	1.14	0	0	12
		Ave: 5	Ave: 1.21	Total:	0	

Among the main group of participants, Participant 13 had the longest average key press length, at 20 ticks, and reported that he found it very difficult to press keys quickly. Participants 10, 12 and 19 also had long average key press lengths. Participant 19 reported that she found short key presses extremely difficult, while Participant 10 also found them difficult. Participant 12 reported no difficulty, having the key repeat facility disabled for his test. He has word processing experience on a PC and reported no problem with long key presses there. It is not known what key repeat setting is in force on his usual machine.

Among the comparison group, the average key press length was 5 ticks, with the longest for any individual being 8 ticks. No key presses longer than 10 ticks were observed.

The standard deviations in key press lengths also reveal differences between the main and control groups. While the average standard deviation among the control group was 1.21, within the main group the average was 4.05. Extreme values were observed for Participants 17 and 18. Both of these participants were novices, and the values recorded were influenced by some abnormally long key presses made when using keyboard commands such as 'command-b'. However, even with these two participants excluded the Mann Whitney U test showed a significant ($p=0.003$) difference between the two groups.

There was a positive correlation ($r=0.387$, $p<0.05$) between the proportion of key presses longer than 16 ticks, and the participants' reported difficulty in pressing keys quickly.

4.5.2 Additional Key Errors

An additional key error is the activation of a key near to an intended key, caused by failure to aim the movement accurately. 276 examples of this type of error were observed in the typing tasks, distributed over 21 of the participants. 97.9% of additional keys pressed were adjacent to the intended key. In 159 of the 276 examples (57.6%), the intended key was also activated, and in 95.5% of these 159 cases, the two key presses overlapped in time. Where more than one key was activated, the intended key was pressed down first in 32% of the cases, the keys were pressed at the same time in 52% of the cases, and the unintended key was pressed first in 16% of the cases. Release of the keys also showed an asymmetrical distribution, with the intended key being raised last in 68% of cases, the keys raised at the same time in 17% of cases, and the unintended key being raised last in 14% of cases.

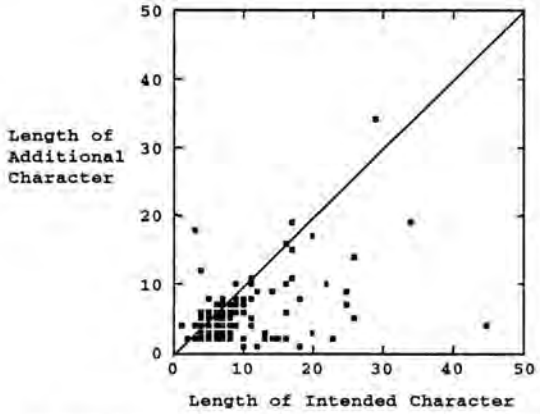


Figure 4.2: Lengths of additional and deliberate key presses I

The lengths (in ticks) of the intended and unintended key presses (excluding errors on modifier keys) are illustrated by the plots in Figures 4.2 and 4.3. In Figure 4.2, the length of each additional key error is plotted against the length of the associated deliberate key press. The line on the graph marks points at which the deliberate and additional key presses were the same length. The figure shows that additional key presses tend to be shorter than the deliberate presses. Figure 4.3 plots keystroke lengths for deliberate and additional key presses, and shows that while unintentional presses tend to be shorter than the deliberate ones with which they are associated, they occupy a similar range of values.

All participants who made more than 5 additional key errors on either task reported some or moderate difficulty in isolating keys or in reaching all the keys on the keyboard. There was a significant correlation (Spearman Rho = 0.528, $p<0.01$) between reported ease of isolating keys, and additional key error rate. Two participants with previous experience reported some difficulty in isolating keys, but made few errors of this type.

For those who make many additional key errors, keyguards are often suggested as a way of reducing these errors. Only one of the participants, Participant 20, usually used a keyguard. For the experiment she used a wrist rest and no keyguard, which she found a more comfortable configuration, of comparable speed to using a keyguard.

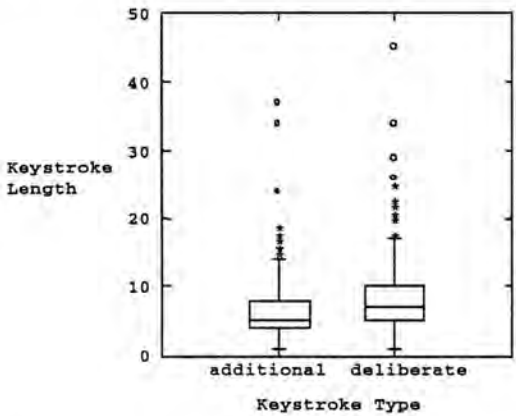


Figure 4.3: Lengths of additional and deliberate key presses II

Participant 1 and Participant 19, who were the most prone to additional key errors, had both tried using keyguards but preferred the keyboard bare. Participant 1 had trouble getting his fingers through the holes, while Participant 19 felt that it would slow her down too much.

Additional key errors were the most common performance error among the comparison group. In both groups, there was a positive correlation (Spearman Rho = 0.523, $p<0.01$, N=26) between experience and error rate.

4.5.3 Missing Key Errors

Seventeen of the participants sometimes failed to activate the key they were aiming for - a total of 179 examples of this error were observed. This was one of the major performance errors for the comparison group, particularly for Participant C5. Missing key errors occur due to lack of pressure on the intended key, or to the movement missing the intended key, perhaps activating a different key. Previous researchers have reported that omission of letters is an important source of spelling errors among touch typists (Grudin, 1983).

Neither of the two participants who made the most missing key errors reported much difficulty in hitting the key they were aiming for, but both of them did report some or moderate difficulty in reaching all the keys on the keyboard. Two experienced participants reported difficulty that was not reflected in their error rates.

4.5.3 Dropping Errors

Dropping errors occur when a user fails to maintain pressure on a modifier key, and it is released before the key to be modified has been pressed down. Table 4.4 (column labelled 'Shift') gives each participant's rating of difficulty of pressing two keys at once, while Table 4.3 shows the number of dropping errors they made, and the number of uses of *Shift* they attempted. 56 examples of dropping errors were observed, among 11 of the participants. Only one of these errors appeared in the comparison group.

Dropping errors are just one manifestation of a larger problem - that of having difficulty in pressing down more than one key at once. Even among non-disabled users, the complexities of the use of modifier keys have been reported to increase entry times and numbers of errors in input tasks (Greenstein and Arnaut, 1987, p.1461). This was the keyboard operation rated as the most difficult by the participants: 3 found it 'impossible', 3 rated it 'hard' or 'very hard', and a further 5 found it 'moderately difficult' or 'quite difficult'.

Of the 28 modified key presses included in the typing task, 17 were capital letters, which could be produced by the use of the *Caps Lock* key. 9 of the participants used this technique to reduce the number of uses of *Shift* required. One participant omitted all capital letters and punctuation entirely. Participant 15 reported that he did not normally use capitals unless absolutely necessary, but did so during the experiment. Partly due to these strategies, and partly due to their efforts to type accurately, not all of the participants who reported difficulty made dropping errors. There was very little correlation (Spearman Rho = -0.113, not statistically significant) between participants' reported difficulty and number of dropping errors made. In fact, the sign of the correlation suggests that those with more difficulty made fewer errors, probably due to the avoidance techniques discussed above.

4.5.4 Bounce Errors

44 examples of bounce errors, as described by Vanderheiden (1992), were observed in 7 participants, all of whom had a motor disability. A bounce error occurs when a key is pressed more than once, perhaps because the user's finger twitched while releasing the key. Most of these 7 participants had an error rate of 1 in 200 or 300 keystrokes. For Participant 6, however, approximately 1 in 70 keystrokes bounced. Participants were not asked how easy they found it to avoid bounce errors.

Table 4.6: Bounce Error and Double Letter Timings

Participant	Bounce Error Rate	Median Bounce Gap (ticks)	Median Double Gap (ticks)	Longest Bounce Gap (ticks)	Shortest Double Gap (ticks)
6	0.0135	4	17	28	2
20	0.0054	4	17	8	2
15	0.0030	4	21	11	2

Table 4.6 provides further detail on the timing and frequency of bounce errors for the three participants most prone to these errors. Error rate is measured as the number of bounce errors per correct character. In all three cases, the median gap between the correct character and a corresponding bounced key press (the median bounce gap) was 4 ticks, while the median gap between deliberately typed double letters (the median double gap), including *Delete* and the arrow keys, was much higher. The participants varied widely in the length of their longest bounce gap, while in all cases the shortest gap between deliberate double letters was 2 ticks - smaller than the median bounce gap value.

4.5.5 Remote Errors

A remote error occurs when a user, while trying to press a key, accidentally presses a different key with a digit or body part other than the one that was intended to touch the keyboard. For example, while reaching to press a key with the first finger, the third finger may drop down and activate a different key. Accidentally pressing down a key by leaning on it is also defined as a remote error, regardless of whether the participant was attempting to make any deliberate key presses at the time. Of the 37 remote errors observed, 25 were on the bottom row of the keyboard. Only Participants 7 and 15

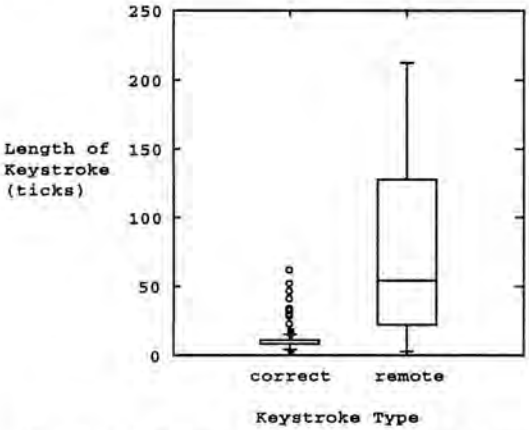


Figure 4.4: Correct and remote keystroke lengths for Participant 7

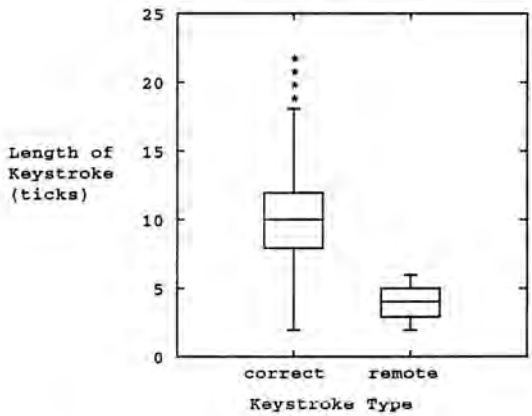


Figure 4.5: Correct and remote keystroke Lengths for Participant 15

seemed particularly prone to remote errors, and Participant 7's total decreased dramatically the second time she performed the typing task.

Figure 4.4 shows two boxplots representing the length of keystrokes for Participant 7's correct and remote key presses. Modifier key presses, and key presses involving errors other than remote errors are excluded. The remote errors are typically much

longer than deliberate key presses. Contrast this with similar box plots for Participant 15, shown in Figure 4.5. For this participant, remote key presses are typically shorter than deliberate, correct key presses.

4.5.6 Transposition Errors

A transposition error occurs when two characters are typed in the wrong order. Transposition errors have been described as a major source of errors in touch typing for both novices and experts (Gentner et al., 1983; Grudin, 1983) and reported as significant in at least one study of spelling errors (Damerau, 1964), although it is not clear whether they occurred through human error or machine malfunction in the latter case. Two examples of transposition errors were observed in the comparison group. However, they were not common among the main group of participants.

Unfortunately, too few examples were observed to allow examination of the relationship between these errors and typing style, typing speed and experience.

It is debatable whether the majority of transposition errors are true performance errors, since they could be attributed to timing misjudgements rather than difficulty in controlling the timing of movements of different fingers. Whatever the definition of transposition errors, they were not an important source of keyboard difficulty for the participants with motor disabilities who participated in this study.

4.6 Mouse Results

Because Participants 1 and 2 performed a different version of the mouse and editing tasks, they are excluded from parts of this analysis, but included wherever possible. Participant 18 is also excluded from this analysis, as he had great difficulty in understanding how to use the mouse and this overrode any physical difficulties he may have had.

Mouse usage data from both the mouse and editing tasks are included in the analysis. Of the twenty-five participants, nineteen performed the mouse task twice, and six only once, due either to fatigue or lack of time. Complete editing tasks, using the mouse, are available for nineteen participants. Partial logs are available for three. Of the

Table 4.7: Summary of mouse difficulties and tasks performed

Par-ticipant	Pointing	Clicking	Multiple clicking	Dragging	Picking up	Time (minutes) M1	M2
1	easy	easy	some diff	easy	easy	n/a	n/a
2	some diff	easy	easy	moderate	impossible	n/a	n/a
3	some diff	easy	easy	moderate	easy	3.0	2.9
4	some diff	easy	easy	some diff	some diff	4.7	3.1
5	some diff	easy	easy	hard	easy	4.6	7.3
6	very hard	some diff	moderate	impossible	moderate	16.6	8.4
7	some diff	some diff	easy	some diff	easy	28.6	14.7
8	some diff	easy	easy	moderate	easy	9.7	.
9	hard	easy	easy	moderate	easy	6.8	5.2
10	easy	some diff	easy	hard	easy	4.7	9.3
11	very hard	moderate	moderate	moderate	moderate	13.7	23.0
12	hard	hard	easy	very hard	easy	14.3	11.1
13	hard	hard	moderate	some diff	easy	21.7	.
14	moderate	easy	easy	extreme	easy	23.0	.
15	moderate	moderate	hard	hard	moderate	10.7	.
16	easy	easy	easy	easy	easy	4.1	3.5
17	easy	easy	easy	hard	easy	16.4	12.6
19	some diff	some diff	very hard	some diff	hard	3.6	2.9
20	hard	easy	easy	hard	easy	15.8	.
C1	easy	easy	easy	easy	easy	7.4	.
C2	easy	easy	easy	easy	easy	4.5	2.6
C3	easy	easy	easy	easy	easy	3.2	2.8
C4	easy	easy	easy	easy	easy	2.7	2.3
C5	easy	easy	easy	easy	easy	3.2	2.4
C6	easy	easy	easy	easy	easy	3.7	2.3

remaining three, two logs were lost due to experimental error, while the sixth was performed using mouse keys, and is excluded from this analysis. Twelve of the participants with disabilities had tried alternatives to the mouse, such as the trackerball. Of these, four preferred to use the trackerball and eight the standard mouse.

All participants were asked how difficult they found it to point, click, multiple click and drag using the mouse. They were also asked how easy it was to pick up the mouse and reposition it on the table. Replies were on a scale ranging from 'easy', through 'some difficulty', 'moderate difficulty', 'hard', 'very hard', 'extreme difficulty' up to 'impossible'. Their responses are given in Table 4.7, along with the minutes taken to perform each of the mouse tasks (M1 and M2).

The times taken by the main group of participants varied from 2.9 to 28.6 minutes. In addition to the effect of physical disabilities, these times are influenced by their experience level and in some cases by cognitive impairments. Nine of the twelve

participants in the main group who performed both mouse tasks performed the task more quickly the second time around, although the difference was not statistically significant (p=0.308, Wilcoxon matched pairs signed ranks test). The comparison group took between 2.3 and 7.4 minutes to perform the tasks. Among this group, the second task was performed significantly faster (p=0.042, Wilcoxon matched pairs signed ranks test).

The difficulties observed in pointing, clicking, multiple clicking and dragging are each discussed in the following sections.

4.6.1 Pointing

Pointing is the most fundamental mouse operation, and yet also one of the most difficult. Seven of the twenty participants with disabilities rated pointing as being as hard or harder than any other mouse operation, and only four rated it as easy.

Unless participants requested otherwise, the most sensitive mouse tracking setting was used. This had the advantage of reducing the range of motion required to operate the mouse. It also reduced the number of times participants needed to lift the mouse and reposition it on the table. One participant found it impossible to lift the mouse and reposition it on the table, while five others reported some difficulty. The disadvantage of using a sensitive tracking setting was that smaller targets were more difficult to pinpoint. Participant 19, who performed the tasks on her own computer, used her usual setting - the third slowest. During the editing task, Participant 11 abandoned use of the mouse altogether and switched to the *Mouse Keys* utility.

The mouse tasks presented required each participant to point to a number of differently sized targets and press the mouse button (either initiating a click or a drag). Many participants had difficulty in pinpointing these targets, or in pressing down the mouse button without moving the mouse off the target. In the main group, the proportion of mouse down events off target ranged from 1.2% up to 47.0% for Participant 20. Eight of the main group of participants had error rates over 20%, and fourteen were over 10%. In contrast, the maximum error rate recorded in the comparison group was 6.3%. Sample sizes ranged from 20 pointing operations up to 266, the average being 119. The Mann Whitney U test showed a significant (p=0.001) difference in pointing accuracy between the main and comparison groups. There was no significant

relationship between experience or reported ease of pointing and pointing accuracy in either the main or comparison group individually. Taking all 26 participants together, however, there was a significant ($r=-0.536$, $p<0.01$) inverse relationship between reported ease of pointing and pointing error rates.

Another indicator of difficulty in pointing may be the time taken to position the mouse. The average time taken by the main group to point and click on the Apple menu was 15.6 seconds (including the time taken to understand the task and locate the target). The comparison group took an average of 6.4 seconds to perform the same task. The average time taken by the main group to point to the word 'so' and start to double click on it was 24.0 seconds, while for the comparison group it was 11.7 seconds.

Examination of the path taken by the mouse in reaching targets showed little difference between the groups. Figure 4.6 shows the mouse path taken by six participants, starting from the top left of the screen, and pointing to a target to the right of the centre of the screen. In both groups, the mouse path was sometimes very direct, sometimes overshot the target, or sometimes took an indirect route to the target.

Another difficulty that some participants experienced was in positioning the mouse without activating the mouse button. Excluding potentially deliberate but randomly placed clicks, 116 unintentional clicks were observed, the maximum for any individual being 19, for Participant 6. Many of these had unwanted side effects such as bringing the Finder to the front or changing the current text view, and recovery could take some time.[2]

4.6.2 Clicking

Difficulty in controlling the mouse can also manifest itself in the mouse clicks themselves. It is particularly interesting to examine any movement between the mouse down and mouse up events. Such movement may or may not cause an error, depending on whether the mouse up event lies on the same target as the mouse down event.

[2] Genuine recovery times are not known, as the observer provided instructions on how to recover from such errors.

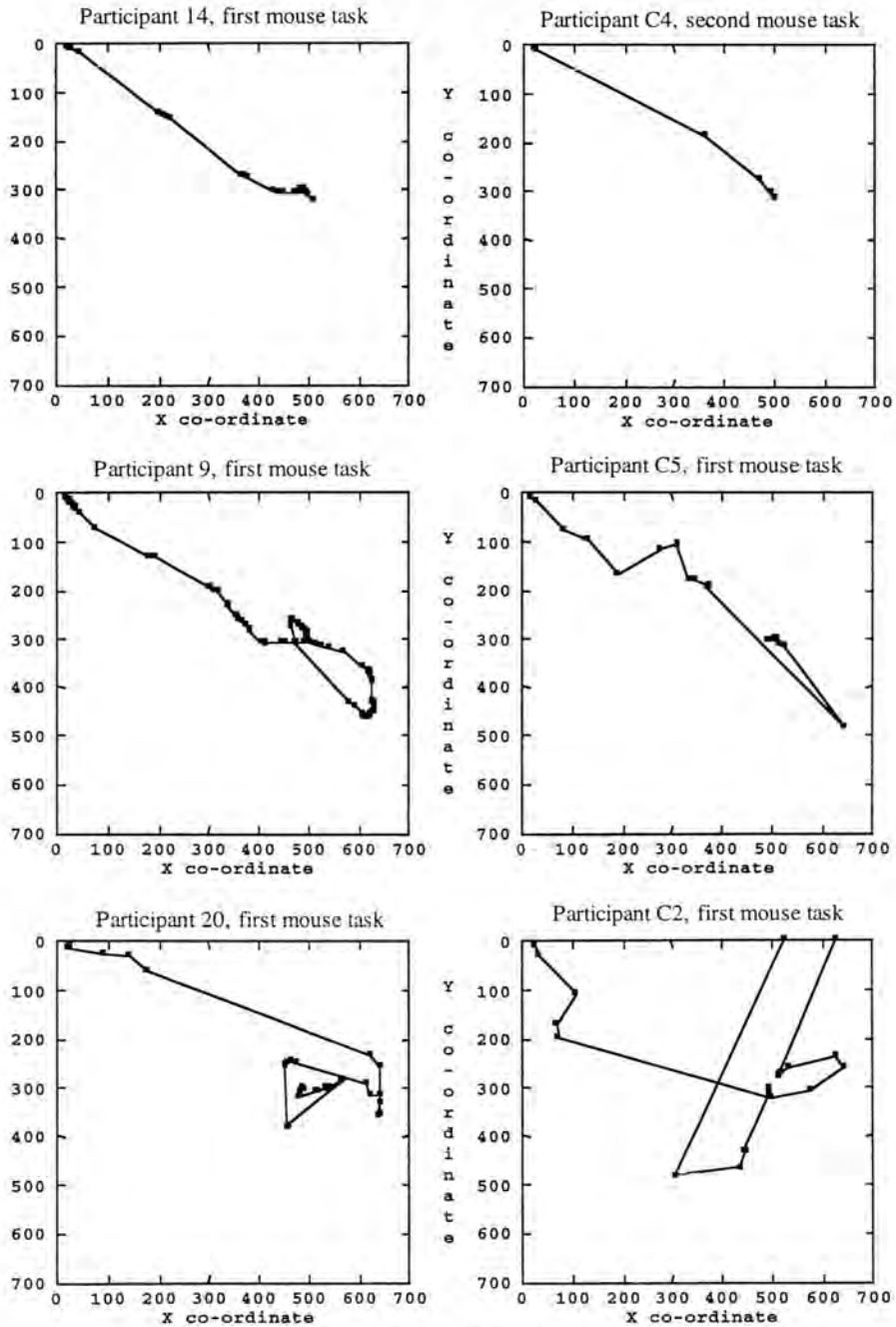


Figure 4.6: Example mouse paths for the same target

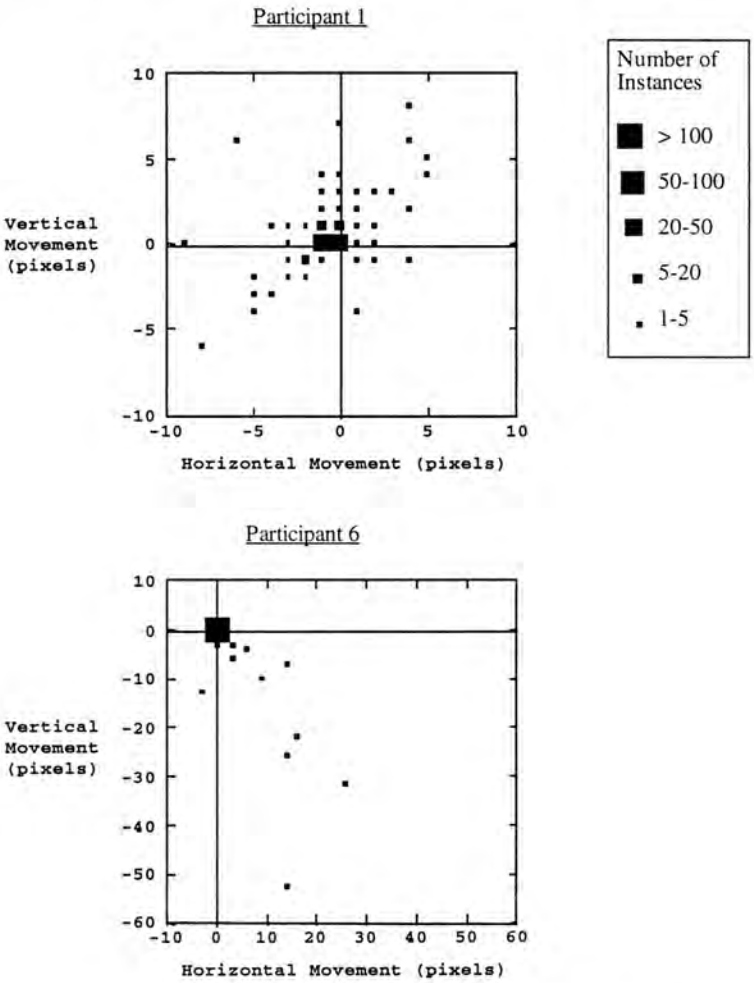


Figure 4.7: Click Movements for Participants 1 and 6

The median value for the percentage of clicks that moved was 28.1% in the main group, compared to 6.3% in the comparison group. The maximum value observed in the main group was 75%, for Participant 1, and the minimum 7.1%, for Participant 3. In the comparison group, the maximum click movement frequency was 15.3%, and the minimum was 1.8%. There was a non-significant (r=0.288, N=25) positive relationship between click movement frequency and the reported difficulty experienced by the participants in performing clicks.

Interestingly, some participants showed strong bias in the direction of mouse movement while clicking. Participants 1 and 6 are good examples. Their click movements are graphed in Figure 4.7. The axes of the graphs measure pixels, and are oriented as they would be on a Macintosh screen. Taking the origin as the mouse down position, the squares show the relative positions at which the mouse was raised, with the size of the square indicating the number of mouse up events at that position.

Participant 1 tended to slip up and to the right or down and to the left. He used his right hand to operate the mouse. Participant 6 slipped down and to the right, and used the mouse in his left hand. Directional biases such as these were observed in 11 participants. The direction of movement appeared to be independent of the hand in which the mouse was held.

4.6 3 Multiple Clicking

For those who find clicking difficult, double, triple or other multiple clicks pose even more problems. The mouse tasks required the user to perform six double clicks, and two triple clicks altogether. The editing task contained at least two double clicks, more if participants chose to use that technique for selecting words in the text.

A total of 164 multiple clicks on a known target were observed. 233 attempts to multiple click on these targets were made, resulting in 141 successful attempts. Of the unsuccessful attempts, 25 missed the target entirely, 29 involved too much movement within or between clicks, and 13 were too slow to be recognised as multiple clicks. In addition, in 23 cases the wrong number of clicks were made.[3] Not all of these were performance errors, sometimes participants deliberately added extra clicks because

[3] The numbers do not add up to a total of 92, because some of the clicks contained more than one error, while other click attempts were unclassified.

earlier clicks had moved, or been too slow, or because the system was slow to respond.

As part of the mouse tasks, participants were also asked to click the mouse quickly many times while keeping it still. This provided additional multiple click data. The average time between connected clicks for each participant in the main group varied between 6 (Participants 10 and 16) and 20 (Participant 6) ticks. The default maximum time between clicks on a Macintosh is 16 ticks. The average for five of the participants in the main group was on or over this value. In the comparison group, average gaps were between 3 and 6 ticks.

Overall, multiple clicking problems were fairly evenly divided between positioning difficulties, difficulties in keeping the mouse still while clicking, and click timing difficulties. There was little or no relation (r=0.025) between participants' reported difficulty in multiple clicking and the accuracy of their multiple clicks.

4.6.4 Dragging

Dragging is a common operation in direct manipulation text editors (Gillan et al., 1990) and other interactive systems (MacKenzie and Buxton, 1994), and the one rated most difficult by 12 of the participants. HCI research has reported dragging to be more difficult and error-prone than pointing (Bewley et al., 1990; MacKenzie, Sellen and Buxton, 1991). This is hardly surprising, since a drag operation requires the ability to point, press the mouse button for a prolonged time, and move the mouse accurately with the button pressed down. Three participants (Participants 6, 11 and 20) found dragging so frustrating that they switched to alternative selection mechanisms. As a result, only a small number of drags were recorded for these participants.

Positioning and pressing down the mouse button at the start of the drag have been discussed in Section 4.6.1. The following discussion examines the drag operations for which the end target of the drag is known. 54 drags deliberately abandoned for reasons such as misunderstanding the task are excluded. Drags abandoned because of physical difficulties are included.

Once a drag had been started, the most common difficulty observed was in raising the mouse button in the correct position at the end of the drag. While 215 correctly completed drags were observed in the main group, in a further 106 examples, a

participant deliberately completed a drag, but the text selected was wrong. Often, they had been careful to position the mouse correctly, but it had slipped as they raised the button.

Another common problem was difficulty in holding down the mouse button while moving the mouse. In 89 cases, the participant accidentally raised the mouse button while dragging.

A third difficulty, of which 38 examples were observed, was that some participants could get stuck and have to abandon a drag. Sometimes this was because they had run out of space to move the mouse and could not lift it up while holding down the mouse button. In other examples, they reached a position where they were physically unable to make the required movement in order to complete the drag.

Finally, 30 drags were abandoned because the participant had moved the pointer off the text window causing it to scroll. This commonly happened with targets at the bottom of the text window.

The comparison group made no dropping errors and did not abandon any drags for physical reasons, or because the screen had scrolled. Of 138 deliberately completed drags, 131 were accurately targeted.

Over the 25 participants considered, there was a significant negative correlation ($r=-0.735$, $p<0.01$) between the level of difficulty reported and the accuracy of drags.

The average time taken to complete a drag varied greatly between participants. The lowest average in the main group was 3.0 seconds for Participant 3, while the highest was 35.3 seconds for Participant 7 (excluding an extreme value of 87.9 seconds for Participant 6, for whom only one successful drag was observed). The median of the averages for the participants in the main group was 9.3 seconds. Among the comparison group, the maximum was 9.5 seconds, the minimum was 2.9 seconds and the median was 3.3 seconds. There was a significant positive correlation ($r=0.435$, $p<0.05$, with the extreme value for Participant 6 excluded) between the difficulty reported in dragging and the time taken to complete drags.

Table 4.8: Participants grouped by disability

Participant	Disability	Keyboard difficulties	Mouse difficulties
12	cerebral palsy	shift, repeat	pointing, clicking, multiple clicking, dragging, avoiding accidental clicks
15	cerebral palsy	shift, repeat, additional, miss	pointing, clicking, multiple clicking, dragging
19	cerebral palsy, RSI	repeat, additional, miss	clicking, multiple clicking, picking up the mouse
20	cerebral palsy	shift, repeat, additional, miss	pointing, clicking, dragging, avoiding accidental clicks
1	stroke	shift, additional	pointing, clicking, dragging
10	stroke	shift, repeat	pointing, clicking, dragging, avoiding accidental clicks
11	stroke/myelitis	shift	pointing, clicking, multiple clicking, dragging
2	incomplete tetraplegia	shift	clicking, dragging, picking up mouse
4	incomplete tetraplegia	shift	clicking, dragging
5	proprioceptive disorder	shift, repeat	pointing, clicking, dragging
6	radial palsy	shift, repeat, bounce, miss	pointing, clicking, dragging, avoiding accidental clicks
3	spasticity, weakness	shift, repeat, additional, miss	dragging
13	co-ordination loss, spasms	shift, repeat	pointing, clicking, multiple clicking, dragging
14	shakiness	shift, repeat	pointing, clicking, dragging
8	muscle wastage	repeat	pointing, clicking, dragging, avoiding accidental clicks
16	muscle loss, weakness		dragging
7	wrist stiffness	repeat, remote, additional	pointing, clicking, dragging
9	shoulder replacement	repeat, miss	pointing, clicking, dragging
17	spina bifida	repeat	pointing, clicking, dragging, avoiding accidental clicks

4.7 Disability and Difficulties Experienced

Table 4.8 shows the difficulties experienced with the keyboard and mouse by each participant. Participants are grouped according to their disability. Participant 18,

whose disability is unknown, and who had difficulty in understanding how to use the mouse, is excluded from the table. A participant is listed as having a specific keyboard difficulty if they had an error rate > 1% in the typing tasks, or if they rated a particular activity as 'hard', 'very hard' or 'impossible'. A mouse difficulty is listed if a participant rated the operation as 'hard', 'very hard' or 'impossible', or if they had an error rate > 10%. The difference in error rates used reflects the fact that mouse operations need not be as accurate as keyboard operations in order to succeed.

Using the same definitions as in Table 4.8, the only difficulties observed among the comparison group were 'shift', for Participant C5, and 'clicking', for Participant C6.

From this table, an impression of the extent to which difficulties are associated with specific disabilities can be gained. For example, all those who had had a stroke experienced difficulty in pressing two keys at once, as did the two incomplete tetraplegics. Those with cerebral palsy all found it useful to alter the key repeat delay. Pointing, clicking and dragging were difficult for the majority of participants, while difficulty in the timing of multiple clicks seemed to be associated with cerebral palsy.

In general, there are similarities in the difficulties experienced by participants with related disabilities. However, difficulties do not appear to be disability-specific. The same problems appear for very different people.

4.8 Discussion

All of the errors and difficulties observed in this small sample are likely to be experienced by many computer users with disabilities. Given the number of participants studied, and the small volume of data examined, there may be additional common difficulties and errors that have not been observed here.

Within both the typing and the mouse tasks, there are likely to be practice effects, particularly for the more novice users. Increasing familiarity with the task should not affect performance errors and is not relevant to this study. Increasing familiarity with the keyboard and mouse may affect the number of performance errors occurring. This study aims to examine performance errors for both experienced and inexperienced users, and so all of the errors and difficulties observed are equally relevant.

The diverse sample of participants presented here illustrates that very different disabilities can cause the same difficulties and errors in the use of keyboards and mice. This is encouraging for developers of generic tools such as those embedded in modern operating systems, which are not developed for people with a specific disability. Studies such as this can provide useful data for developers of such tools.

4.8.1 Keyboard Usage

A set of seven keyboard error types has been identified, six of which were important for the main group of participants. This classification offers a potential framework for description of a user's keyboard errors, and for specification of the errors that can be eliminated or reduced by a given keyboard configuration facility. Errors are, however, not the only factor to be considered. Input rate and the effort required to operate the keyboard are also very important. In keyboard configuration, this framework could be used along with measures of input rate and ease of use, in order to identify facilities that may be relevant to a given user.

In using the keyboard, the main group of participants rated use of the *Shift* key as being particularly problematic. Aiming at a specific key and pressing only the intended key were also difficult for many participants. Most participants found it easy to press keys quickly, but some had extreme difficulty. None of the participants had great difficulty in reaching all of the keys on the keyboard. It may be that such users tend to use devices other than keyboards for input.

In general, there was significant correlation between participants' error rates and reported ease of performance of tasks, although the correlation coefficient was not always high. For some experienced users, difficulties were reported but not observed in the typing samples. This may be because the experts' opinions were based on long term experience, and the typing samples were too small to illustrate their errors. An alternative explanation is that experts may have developed effective techniques for avoiding errors, while still experiencing difficulty in executing the required movements.

Six classes of performance error were notable in this study: long key press errors, additional key errors, missing key errors, dropping errors, bounce errors and remote errors. All of these error types, with the exception of missing key errors, have been

reported in the literature, as described in Chapter 2. While the comparison group spent an average of 1.7% of their time correcting errors of this kind in the second typing task, the main group spent an average of 7.3% of their time on these corrections. Performance errors are clearly important.

In this investigation, the majority of potential long key press errors were suppressed. It is possible that some of the participants would have pressed the keys more quickly had a shorter key repeat delay been in force, resulting in a smaller number of errors than that predicted. Chapter 5 presents evidence that some people with longer 'natural' key press lengths do adjust their key press lengths to some extent, but all remain prone to long key press errors. The data recorded suggest that disability can have a significant effect on both key press lengths, and the variation in key press lengths. The variations both within and between individuals, and the high upper values found indicate that key repeat settings are perhaps the most important issue in keyboard accessibility.

Examination of the keystroke timings for these errors suggests that long key press errors, additional key errors and bounce errors may be automatically recognisable. Missing key errors are unlikely to be recognisable. Dropping errors, and difficulty in the use of *Shift* in general, may be best identified through keystroke patterns, including those techniques users employ in order to avoid the use of *Shift*. Remote errors can have very different timing patterns for different users. This may be because some are caused by activating keys while making key presses, while others are caused by leaning on parts of the keyboard for relatively long periods of time. The latter may be automatically recognisable. More data would be required in order to fully analyse the former.

4.8 2 Mouse Performance Errors

The majority of the participants with disabilities found the mouse difficult to use, and many generally try to avoid mouse operations as much as possible. Pointing and dragging were the most difficult operations, and high error rates were observed in the main group. For pointing and dragging there was a significant correlation between errors occurring and difficulties experienced, indicating that errors, if detectable, could be used to identify users having difficulty in using the mouse.

Independently of error rates, the main group also took longer to perform positioning and dragging operations. It should be noted, however, that the times given include the time taken to acquire the next task (either by reading it from a sheet, or having the observer read it out), understand the task, and locate the target on the screen. For some participants, cognitive impairments had an impact on the time required for these tasks, so the difference in times between the main and comparison groups cannot be solely attributed to the effects of physical disabilities.

Participants also experienced difficulty in holding the mouse still while clicking, in performing multiple clicks quickly, and in repositioning the mouse on the table. Reported difficulties in clicking and multiple clicking did not correlate well with movement within clicks and movement within or errors in the timing of multiple clicks. Many participants showed directional biases in the movements made within clicks. This is an important finding. A mouse configuration tool could potentially use such biases to help to discriminate between clicks which moved and small drag movements, thereby providing a system which has some tolerance to mouse movement within clicks.

Compared with keyboard usage, automatic recognition of mouse difficulties in the absence of knowledge about the user's task is much more difficult. It would require a mechanism to differentiate between accidental and deliberate clicks, between clicks which moved and small drags, and between multiple clicks and single clicks closely spaced in time.

Despite problems with the mouse, most participants who had tried alternative pointing devices such as the trackerball preferred the standard mouse, in some cases because they were more used to it. For some people, the convenience of using the default input devices outweighs the frustration caused by difficulties in using them.

4.9 Summary

Very little detailed information about the usage of input devices by people with motor disabilities has been published. Such data is essential in order to ascertain the range of difficulties encountered, and to establish how they are manifested in the input stream.

This chapter has presented a study of twenty keyboard and mouse users with motor disabilities, and compared them with six non-disabled users. Six keyboard performance errors have been identified, some of which may be amenable to automatic

recognition. Mouse difficulties were greater than keyboard difficulties, and extended to every aspect of mouse usage. In the absence of task knowledge, they are likely to be more difficult to recognise than keyboard errors.

The findings of this study are important to designers of input devices, configuration facilities, and mainstream applications. They highlight existing areas of difficulty with standard input devices and give some impression of the relative difficulty of specific input operations such as pointing and clicking.

On both the keyboard and mouse, error rates did not necessarily reflect the level of difficulty reported by participants. Some people are unaware of their errors, or do not consider them a problem, while others may successfully avoid making errors, at the expense of time or effort.

Chapter 5 presents further empirical evidence of the importance of long key press errors, and difficulties in the use of modifier keys. It highlights the potentially crucial role that keyboard configuration facilities, in this case *Repeat Keys* and *Sticky Keys*, can play in reducing errors and improving keyboard usability.

Using the data gathered in this study and presented in Chapter 5, Chapter 6 will then examine the coverage of the existing keyboard and mouse software configuration facilities. This will relate the facilities described in Chapter 2 to the data from this study, and examine the extent to which the configuration facilities are used by those who may benefit from them.

Chapter 5

The Potential Impact of Repeat Keys and Sticky Keys on Keyboard Usability

The two best known, and probably most widely used, keyboard access facilities are *Sticky Keys* and *Repeat Keys*. While the value of these facilities has long been recognised, there has been very little academic research examining their use. Here, empirical data on the use of these two utilities by people with and without motor disabilities is presented. The utilities are examined in terms of error reduction capabilities and user preferences. This examination provides empirical evidence of the potential of access facilities to improve keyboard access by reducing both input errors and the effort required to operate the keyboard.

The data described here was gathered during the evaluation study described in Chapter 8. As a side effect of the evaluation, data on the usage of *Sticky Keys*, *Repeat Keys*, and one other experimental configuration utility were gathered. The additional, experimental utility is described in Chapter 6, Section 6.5.

5.1 Participants

Thirty unpaid volunteers took part in the study. Twenty had some disability affecting their typing (participants ED1-ED20), while ten had no relevant disability (participants EN21-EN30). Table 5.1 summarises the participants' previous computer experience, the effect of their disability on keyboard use, and their typing style (left or right hand only, left or right hand preferred, both hands, or touch typing).

Table 5.1: Participant summary

Participant	Experience (years)	Disability affecting typing	Typing style
ED1	34.0	constant movement and spasms in the hands and fingers	left
ED2	0.4	difficulty in controlling hand and finger movement without visual attention	left
ED3	28.0	pain when making keystrokes, constant movement and occasional spasms	both
ED4	7.0	control of hand movement requires effort, movement slow	right
ED5	0.2	loss of dexterity in hands and wrists, movement stiff and painful	both
ED6	2.0	no use of left hand, difficulty in making precise, aimed movements	right
ED7	1.5	loss of dexterity in left hand	mainly right
ED8	0.0	movement control and force difficulties, some spasm	both
ED9	0.1	pronounced hand shake, particularly left hand	both
ED10	5.0	difficulty and pain when moving wrists and fingers	both
ED11	28.0	difficulty in straightening fingers, some hand spasms	both
ED12	0.2	spasm in hands and arms, particularly when fatigued	both
ED13	29.0	impaired hand dexterity	mainly right
ED14	0.04	tremor and spasms in hands	mainly left
ED15	2.0	right hand unsteady and shaky	mainly left
ED16	1.02	right hand unusable	left
ED17	6.0	co-ordination difficulties, muscle spasms	both
ED18	7.0	semi-paralysis on left side and weakness on right, very painful to touch any object	mainly right
ED19	0.8	limited use of left hand, difficulty flexing fingers of right hand	mainly left
ED20	12.0	difficulty in controlling movement and spasm in hands	both
EN21	2.4	none	both
EN22	9.0	none	touch
EN23	0.3	none	both
EN24	14.0	none	both
EN25	2.8	none	touch
EN26	18.0	none	both
EN27	0.0	none	mainly right
EN28	7.0	none	both
EN29	22.0	none	touch
EN30	0.0	none	both

Disabilities affecting use of the keyboard included cerebral palsy, effects of stroke, spasms, nerve damage, incomplete tetraplegia, multiple sclerosis and arthritis. The

effects of these disabilities on keyboard use included tremor and spasm in the hands and fingers, co-ordination difficulties, loss of dexterity, weakness and pain when pressing keys.

There were seventeen men and thirteen women, aged between 19 and 73. The average age of the people with disabilities was 47, while the average age for those with no disabilities was 38. The keyboard experience of the participants ranged from none at all to 37 years of daily use. The mean number of years of experience of those with a disability was 8.21, while for those with no disability it was 7.55. There was no significant difference between the disabled and non-disabled groups in terms of age ($t=-1.669$, $p=0.106$) or experience level ($t=-0.162$, $p=0.872$). Experience was measured as the number of years of daily computer use each participant had had, where daily was considered to be five days a week or more. For participants who used computers less frequently, a daily use figure equivalent to their weekly use was calculated. For example, five years of weekly computer use was considered to be equivalent to one year of daily use.

Three of the participants, all with no disability, were non-expert touch typists. Of the remainder, twelve typed wholly or mainly with one hand, one using an artificial prodder held in one fist, while fifteen used both hands. Unless otherwise mentioned, the participants did not usually use *Sticky Keys* and had not altered the key repeat delay on their usual computer from the default setting.

- Participant ED1 was a forty-year old man with cerebral palsy, with 34 years of daily computer experience using PCs and Sun workstations. He typed with his left hand only, usually used a key repeat delay longer than the default, and sometimes used *Sticky Keys*, depending on what he was doing.

- Participant ED2 was a seventy-two year old woman with proprioceptive difficulties due to peripheral nervous system damage as a result of chemotherapy and exposure to radiation. She had been using Macintosh computers once a week for two years, in a class targeted at people with disabilities. She reported that she used to find a longer key repeat delay and *Sticky Keys* useful but didn't tend to use them anymore. She typed with one finger of her left hand only.

- Participant ED3 was a thirty-three year old woman with cerebral palsy and RSI She had 28 years of daily computer experience on Macintosh and PC platforms, and most often used a Macintosh. She typed with both hands, but sometimes used

Sticky Keys on a portable computer. She also used a slowed key repeat delay of 24 ticks.

- Participant ED4 was a thirty-eight year old man with cerebral palsy. He used PCs daily at home and for voluntary work with a local charity, and had seven years of computer experience. He typed predominantly with his right hand, always used *Sticky Keys* when it (or a similar utility) was available, and preferred to use a key repeat delay longer than the default.

- Participant ED5 was a sixty-two year old man with multiple sclerosis. He had been using a PC once a week for three months, and had a year of similar experience two years previously. He typed with both hands, found *Sticky Keys* useful, and did not know what key repeat delay he would prefer.

- Participant ED6 was a sixty-three year old man whose disability stemmed from a major stroke. He had no use of his left arm and hand, and tended to find visual focusing and aim difficult with the keyboard. He had ten years of computer and word processing experience, gained before and after his stroke. His computer usage consisted of a weekly session as a volunteer worker for a charity. There he used a Macintosh with *Sticky Keys* activated.

- Participant ED7 was a thirty-nine year old man who, due to an accident and subsequent operation, had difficulty extending the fingers of his left hand, being confined largely to pincer-like movements. He typed with his right hand and left thumb.

- Participant ED8 was a thirty-two year old man who experienced difficulty in controlling his movements and applying pressure (particularly to typewriter keys). He had used a PC and BBC Microcomputer twenty years previously, but almost never used computers at the time of the experiment, and could not remember when he had last regularly used a keyboard. He typed with both hands.

- Participant ED9 was a sixty-three year old woman who had a nervous disability causing her hands to shake, particularly her left hand. She had used a Macintosh eight years previously for a few weeks, but had no other computer experience. She typed with both hands.

- Participant ED10 was a thirty-four year old woman with rheumatoid arthritis. She had twenty years of typewriter and keyboard experience, and used PCs at home and at college once or twice a week, with an ergorest supporting one or (preferably) both her forearms. She typed with both hands.

- Participant ED11 was a forty-six year old man with neurological damage resulting in limited finger dexterity. He used PC and sometimes Macintosh computers daily, and had done so for twenty-eight years. He used a splint to steady the first finger of his right hand and typed with both hands

- Participant ED12 was a seventy-three year old woman who had been using a PC every day for three months. She had cancer of the thyroid, with growths under her fingernail and on her wrist and forearm. Her main difficulty in typing was spasm in her arms and hands, which increased when she was tired. She typed with both hands.

- Participant ED13 was a forty-eight year old man with incomplete tetraplegia as a result of an early rugby accident, which impaired dexterity in his hands. He had been using computers for 29 years. He usually used a PC daily , but also used Macintosh and UNIX platforms. He typed predominantly with his right hand, but used his left for control keys. He used a key repeat setting longer than the default.

- Participant ED14 was a forty-one year old man with a spastic condition who typed mainly with his left hand, using his right for modifier keys. He had been using a PC three times a week for the previous four weeks.

- Participant ED15 was a fifty-six year old woman. Cerebral haemorrage and ataxia had affected co-ordination on her right hand side, and she typed mainly with her left hand. Six years previously she had used an IBM mainframe computer daily for two years, but after becoming disabled four years ago had not used one.

- Participant ED16 was a thirty-four year old woman who had had a stroke nine months previously, which had predominantly affected her right side. She typed with her left hand only, and for the previous month had been using a PC with a one-handed Maltron keyboard twice a week. Twelve years previously, she had used computers for a year at university.

- Participant ED17 was a fifty year old man. He experienced spasms and difficulty in co-ordination, caused by having been prescribed the wrong medication for epilepsy. He had been using computers daily for six years, and was undertaking a word processing course, using PCs running Windows 3.11. He typed using both hands.

- Participant ED18 was a forty-two year old man with a damaged central spinal cord resulting in semi-paralysis on his left side and weakness on his right side. He found it painful to touch any object, including computer keys, but used a PC daily, and had done so for the past seven years. He used a repeat delay slower than the default and typed mainly with his right hand, using his left for modifier keys.

- Participant ED19 was a sixty-one year old man with incomplete tetraplegia, who had difficulty flexing his fingers. He usually used a PC running Windows 95 at home for word processing, about once a week, and had four years of computer experience. He typed using a prodder held in his right hand, and usually used *Sticky Keys*, but could also use his left hand with some effort. He usually used a longer than default repeat delay setting, for the benefit of his (not disabled) son rather than himself.

- Participant ED20 was a twenty-one year old woman with cerebral palsy who had been using PCs daily for twelve years. She typed with both hands, and preferred to use a tilted keyboard. She used a repeat delay longer than the default.

- Participant EN21 was a non-disabled twenty-one year old man. At the time of the experiment he used computers once a month, but had used a Macintosh daily for two years at college, and an IBM PC weekly for two years while working. He typed with both hands.

- Participant EN22 was a thirty year old woman with no relevant disability. She used a PC every day and had nine years of PC and typewriting experience. She was a non-expert touch typist, typing at a rate of approximately 29 words per minute.[1]

[1] A word is defined as five characters, spaces are ignored.

- Participant EN23 was a nineteen year old woman with no disability. She typed with both hands. She had been using Macintosh and PC computers weekly for 1.5 years, while at university.

- Participant EN24 was a twenty-nine year old man with fourteen years of computer experience. He had no disability. He used a Silicon Graphics workstation daily at work, and typed with both hands.

- Participant EN25 was a twenty-nine year old man with no physical disability. he had been using Macintosh computers several times a week for four years while at university. He was a novice touch typist, typing at a rate of approximately 26 words per minute, and preferred to use a longer key repeat delay than the default.

- Participant EN26 was a thirty-six year old woman who used PCs daily at home and at work, and had been using computers for eighteen years. She typed with both hands, usually used a wrist rest, and had no disability.

- Participant EN27 was a sixty year old woman who had no disability and had never used a computer before. During the experiment she typed mainly with one finger of her right hand.

- Participant EN28 was a fifty-three year old woman who had no disability, and had been using computers for thirty-seven years. She used a PC once a week and typed with both hands.

- Participant EN29 was a forty-six year old man who had been using computers daily for twenty-two years. He usually used PC and UNIX machines. He had no disability, and touch typed at a rate of approximately 22 words per minute

- Participant EN30 was a fifty-two year old man with no disability who never, or only very rarely used a computer. He typed with both hands.

Participants with disabilities were recruited through personal contacts (11), advertisement in the local press (6), and through organisations providing computing for people with disabilities (3). Those with no disability were recruited through personal contacts.

5.2 Materials

Four matched text passages were used. These are reproduced in Appendix A.2. Each required 625 keystrokes, and included 21 capital letters and 9 punctuation marks requiring the use of the *Shift* key. Each passage contained one sequence of four capital letters for which use of the *Caps Lock* key was appropriate (although not essential).

Macintosh 475 8/160 and Power Macintosh 6100/66 machines, and the SimpleText© word processor were used.

On the keyboards used in the experiment, the *Caps Lock* key could be used to generate capital letters, but not punctuation marks which normally required the use of *Shift*.

5.3 Procedure

Experimental sessions were limited to two hours, and extended only if the participant chose to continue. Participants were informed that they were free to stop or rest at any time.

Background information about the participant, including their level of keyboard experience and previous knowledge of *Repeat Keys* and *Sticky Keys* was recorded. Participants were asked: "How useful is this facility to you?" for both *Repeat Keys* and *Sticky Keys*. Responses were given on the scale: 'not useful', 'somewhat useful', 'useful', 'very useful', or 'essential'. For *Repeat Keys* they were asked to give their preferred setting for the key repeat delay, if known. Participants who had not previously encountered these facilities were given a verbal description and if necessary a demonstration, but were not given the opportunity to try the facility before responding.

Participants were then asked to copy one of the passages as accurately as possible using the default keyboard configuration. They were free to make corrections if they wished, or to ignore their errors. Negative transfer of learning effects are to be expected for participants used to a different configuration, and these effects in this and other typed passages will be identified and discussed at various points in this analysis.

Depending on the keyboard difficulties observed during this task, the participants were then asked to copy up to three further passages, each with a particular access facility

enabled. Ideally the sequence in which the facilities were enabled and disabled would have been varied between participants. Unfortunately, this was not possible, due to constraints imposed by the evaluation study. As a result, *Repeat Keys* was most often the first facility considered, and *Sticky Keys* was most often the last.

In examination of *Repeat Keys*, the key repeat delay was altered, with the participant's knowledge, to suit their typing style. When increasing the key repeat delay, the choice of setting for the new delay was made using the participant's own opinion, or a suggested value based on the length of key presses made in the first passage, chosen by the model described in Chapter 7. The ideal value is one which eliminates long key press errors while still allowing keys to repeat as fast as possible, enabling activities such as document navigation and multiple deletions to take advantage of the key repeat facility.

Participants were asked to copy a different passage with the new keyboard configuration, and were again asked for their opinion on the usefulness of the facility, using the same scale as previously.

When *Sticky Keys* was activated, participants were told that they should generate capital letters by pressing and releasing *Shift* exactly once, and then pressing the desired character, or by using *Caps Lock* if preferred. They were not instructed on the more advanced features of *Sticky Keys*, such as the locking mechanism. Their attention was not drawn to the visual feedback *Sticky Keys* provides on the current status of *Shift*. They were not required to activate the utility themselves.

The intention here is to examine the use of the alternative mechanism for generating modified characters, provided by *Sticky Keys*, and to note any difficulties in the use of this mechanism. The usability of the utility as a whole requires further research.

Participants were asked to type a passage using the *Sticky Keys* facility, with the key repeat delay and all other settings at the default value as used in the first passage. Again, participants were then asked for their opinion on the facility and its relevance to them.

Ideally, participants would also have been asked to copy a final passage with the default configuration, in order to allow measurement of practice and fatigue effects. It would also have been useful to examine the combined use of the utilities. In practice this was not feasible. Many participants did not complete all four passages due to fatigue or time constraints. A further passage would have extended the experiment

time beyond the acceptable limit for many of the participants and may have deterred them from volunteering. All participants typed passages in a single session, with breaks between passages, with the exception of Participant ED8, who typed two passages on two separate sessions, having become too tired to continue after the initial passage on the first session. Where more than two passages were typed, some measurement of practice effects is possible, and is described later in this chapter.

The order in which the text passages were presented was varied between participants, in order to counteract any passage-specific effects.

5.4 Data

For each participant, the following data was recorded:

- *An automatically generated log of input events*, including time stamped key down and key up events.
- *A video of the participant's hands typing* at the keyboard. This is used to help distinguish between difficulties in use of modifier keys, and errors due to other causes.
- *Observations made during the tasks.* For each participant, the same observer (the author) recorded impressions of the difficulties experienced by the user, and noted examples of long key press errors, dropping errors, difficulties in using modifier keys, and other relevant events.
- *Background information about the participant.* This included details of their previous keyboard experience, and their previous awareness of *Sticky Keys* and *Repeat Keys.*
- *The participant's opinion on the facilities used* for each access facility tried. The opinions of the participant on the relevance of the facility and its usefulness to them were recorded both before and after trying each facility.

5.5 Analysis

The observations made and video evidence were used to manually annotate the recorded log files, indicating dropping errors made and time spent correcting dropping and long key press errors. This process required differentiation between dropping errors and cases where a participant who usually uses *Sticky Keys* deliberately released *Shift* before pressing down the key to be modified. Any observed difficulties in using the utilities were also annotated.

The annotated log files were then automatically filtered and the Systat statistical package (SYSTAT, 1992) used to perform the analyses. Nonparametric statistics were used, as the variables under examination do not have, or cannot be assumed to have, normal distributions.

5.6 Results - Repeat Keys

5.6.1 Long Key Press Errors

A long key press error is a key press intended to generate x characters (usually x=1), which actually generates x+y (y>0) characters due to the key repeat facility. It is difficult to accurately identify long key press errors occurring on the *Delete* key, or the arrow keys, where x may be greater than one. As a result, where this chapter presents error frequency values, these indicate the percentage of key presses other than modifier keys, *Delete*, and the arrow keys which were long key press errors.

In the first passage typed all participants used the default key repeat delay of 16 ticks. Sample sizes were in the range 568-761 keystrokes. With the default delay, nineteen of the thirty participants made long key press errors, and these are summarised in Table 5.2. The table shows all of the participants who made errors, the number of years of computer experience they had, their long key press error rate, measured as the percentage of alphanumeric characters causing an error, and the percentage of their total time that was spent correcting errors of this type.

Table 5.2: Long key press errors in the initial passage

Participant	Experience (years)	% errors	% time correcting errors
ED1	34.0	6.4	2.9
ED2	0.4	4.7	4.7
ED3	28.0	18.9	37.6
ED4	7.0	14.0	21.4
ED5	0.2	4.9	3.8
ED6	2.0	2.1	0.4
ED8	0.0	21.0	31.3
ED9	0.1	2.1	1.5
ED10	5.0	0.2	0.0
ED11	28.0	1.1	9.0
ED12	0.2	1.4	4.5
ED14	0.04	1.2	0.7
ED15	2.0	2.3	1.9
ED16	1.0	0.5	0.2
ED17	6.0	74.4	44.8
ED20	12.0	0.6	1.2
EN21	2.4	0.2	1.4
EN22	9.0	0.2	0.0
EN27	0.0	0.3	0.0

Sixteen of the twenty participants with disabilities made long key press errors, while only three of the ten with no disability made errors. Error rates ranged from 0% up to 74.4%, and the four participants with error rates greater than 10% - Participants D3, D4, D8 and D17 - accounted for 81.7% of the long key press errors observed. The four participants who made the most errors of this type all spent over 20% of their time correcting them. The highest error rate observed for a participant with no disability was 0.3%.

There was no significant correlation between experience and error rate in either the disabled or non-disabled group.

Some of the errors observed were due to negative transfer of learning effects, where participants were used to a longer key repeat delay. Comments from participants with high error rates included:

> "This is far more sensitive ... this is like working on a PC." (Participant ED3)
>
> "This doesn't normally happen to me. Have you changed the settings?" (Participant ED8)
>
> "I'm getting frustrated!" (Participant ED17)

The very high error rates observed suggest that these participants were unable to adjust to the shorter delay. This confirms the findings of Chapter 4: long key press errors can represent a serious barrier to keyboard usability.

The nineteen participants who made long key press errors could potentially benefit from using *Repeat Keys* to extend the key repeat delay, while the remaining eleven are potential users of a delay shorter than the default value. Of the former group, ten went on to try a longer key repeat delay, while ten of the latter group tried a shorter delay. The results are reported in Sections 5.6.4 and 5.6.5 respectively.

5.6.2 Fatigue Effects

There may be influences other than the key repeat delay affecting long key press error rates. Increases in a user's key press lengths over time could increase the error rate. Here, this is referred to as a *fatigue effect*, under the assumption that fatigue can make it more difficult to press keys quickly.

Fatigue effects were examined using the Mann-Whitney U test to check for significant ($p<0.05$) increases in key press lengths between passages for each participant, using all the passages they typed. As before, modifier keys, the *Delete* key, and the arrow keys were excluded. In order to eliminate potential interference with passages typed using *Sticky Keys*, modified keys were also excluded. Some passages were typed using a facility which affected the keyboard's response to adjacent overlapping keystrokes (described in Chapter 6). It has been assumed that the use of this facility has no effect on key press lengths.

A significant increase in key press lengths over time was found for eight participants (ED5, ED9, ED10, ED12, ED16, ED17, ED18 and ED20). All eight had a disability affecting their typing. One of these participants (ED17) typed only two passages, the second using an altered key repeat delay, so the observed change could be due to the effect of the altered key repeat delay. For the remaining seven, however, the change cannot be attributed to the use of *Repeat Keys*, since all either typed a further passage after having used an altered key repeat delay, or did not use an altered delay in any passage.

5.6.3 Practice Effects

In opposition to fatigue effects, *practice effects* may have acted to reduce a participant's key press lengths over time as they became accustomed to the keyboard and familiar with the repeat delay in force, and adjusted their keystroke lengths accordingly. If their error rate was initially non-zero, practice effects could have reduced their error rate over time.

Of the thirty participants, eight exhibited a significant reduction in key press length over time - Participants ED4, ED6, ED7, ED11, ED15, EN21, EN23 and EN27. Two were novice keyboard users, three were occasional users, and three usually used a PC platform with a key repeat delay longer than 16 ticks. The two most extreme examples are Participant ED4, whose error rate was 14.0% in the first passage typed, and 6.7% in the final passage typed, and Participant ED11 whose error rates were 1.1% and 0.3% in the first and final passages typed.

5.6.4 Increasing the Key Repeat Delay

Ten people (nine with some motor disability) tried a delay longer than the default setting of 16 ticks. All had non-zero error rates in the initial passage. Participant ED2 chose to disable key repeats altogether. For the remaining participants the delay used was determined by the model described in Chapter 8, which uses information about the key press lengths in the initial passage to choose an 'ideal' delay value. An ideal value is one which minimises long key press errors while still allowing keys to repeat as quickly as possible. The actual delay imposed was the nearest available setting at or above the model's recommendation. Three participants used a delay of 40 ticks, and six used a delay of 24 ticks.

All participants completed the passage with the exception of Participant ED17, who copied 2/3 of the passage, due to fatigue and time limitations. The sample size recorded was therefore smaller, at 332 keystrokes.

Table 5.3: Increasing the key repeat delay

Participant	Delay tried (ticks)	% errors with default delay	% errors with new delay	% time correcting errors with default delay	% time correcting errors with new delay	Significant changes in key press length between passages
ED2	off	4.7	0.0	4.7	0.0	none
ED3	40	18.9	0.0	37.6	0.1	none
ED4	24	14.0	0.2	21.4	0.2	decrease
ED8	40	21.0	0.0	31.3	0.0	(none)
ED9	24	2.1	0.3	1.5	0.2	increase
ED11	24	1.1	0.0	9.0	0.0	decrease
ED12	24	1.4	0.6	4.5	2.2	increase
ED17	40	74.4	0.3	44.8	0.0	(increase)
ED20	24	0.6	0.0	1.2	0.0	increase
EN22	24	0.2	0.0	0.0	0.0	none

Table 5.3 shows, for each of these participants, the delay value they used, their initial error rate, and their error rate under the new delay. It also shows the percentage of their time the participants spent correcting long key press errors in each condition. The final column of the table shows whether the participant's key press lengths were changing significantly over time, irrespective of use of *Repeat Keys*. Entries in brackets indicate that the participant typed only two passages. For these participants, changes over time could therefore be attributable to the use of *Repeat Keys*, or to practice or fatigue effects.

Error rates with the new delay ranged from 0% up to 0.6%, and the maximum time spent correcting errors was 2.2%. Only two of the ten participants showed a decrease in key press lengths over time, and so for the remainder the error reduction can be clearly attributed to the increased key repeat delay.

The four participants with original error rates over 10%, who accounted for 81.7% of the errors in the original passage, had error rates of 0%, 0.2%, 0% and 0.3% using the increased delays, and spent up to 0.2% of their time correcting these errors. Comments from these participants included:

> "It's amazing the difference" (Participant ED3)
>
> "This feels a lot better" (Participant ED4)

For two participants, there appeared to be some effect of increasing the key repeat delay on key press lengths. When the repeat delay was increased, the Mann Whitney

U test showed a significant ($p<0.05$) increase in key press lengths for Participants ED3 and ED17. The strongest effect observed was for Participant ED17, for whom 93.1% of key presses were 16 ticks or longer when the key repeat delay was set to 40 ticks, as opposed to 74.4% when the repeat delay of 16 ticks was in force. As mentioned previously, Participant ED17 only copied two passages, so it is not known whether this effect was due to *Repeat Keys* or fatigue effects. However, Participant D3 typed three passages. There was no significant difference ($p=0.569$, Mann Whitney U test) between her key press lengths in the first and third passages. While using an increased key repeat delay in the second passage, her key press lengths were significantly longer than those in the first and third passages ($p=0.003$ and $p=0.028$ respectively, Mann Whitney U test). Percentages of alphanumeric key presses 16 ticks or longer were 18.9%, 23.6% and 14.7% in the first, second and third passages respectively.

5.6.5 Decreasing the Key Repeat Delay

Ten participants who had no long key press errors in the initial passage went on to try a shorter key repeat delay. On the Macintosh, the only available delay setting shorter than 16 ticks is 12 ticks, and this setting was used for all ten. Three of this group had a disability affecting their typing.

Table 5.4 shows the effect of reducing the key repeat delay on error rates and time spent correcting errors for these ten participants. Also shown is any change in their key press lengths over time. Entries in brackets indicate that the participant typed only two passages.

Long key press errors were observed for four of the ten participants, three of whom had no disability. The error rates observed were low, suggesting that these participants had little difficulty in making key presses less than 12 ticks in duration. It is likely that participants whose key press lengths sometimes exceeded 12 ticks were able to make shorter key presses to compensate for the shorter delay. That there were errors, however, suggests that the shorter delay was less comfortable for some participants than the 16 tick delay. Reducing the key repeat delay too far can introduce errors regardless of disability.

Table 5.4: Decreasing the key repeat delay

Participant	Delay tried (ticks)	% errors with default delay	% errors with new delay	% time correcting errors with default delay	% time correcting errors with new delay	Significant changes in key press length over time
ED13	12	0.0	0.0	0.0	0.0	none
ED18	12	0.0	0.8	0.0	0.0	increase
ED19	12	0.0	0.0	0.0	0.0	none
EN23	12	0.0	1.0	0.0	8.1	decrease
EN24	12	0.0	0.0	0.0	0.0	none
EN25	12	0.0	0.4	0.0	0.5	none
EN26	12	0.0	0.0	0.0	0.0	none
EN28	12	0.0	0.0	0.0	0.0	none
EN29	12	0.0	0.0	0.0	0.0	none
EN30	12	0.0	0.2	0.0	1.2	(none)

5.6.6 User Preferences

This section examines the actual usage of the *Repeat Keys* facility among the participants. Prior to the experiment, ten of the participants with motor disabilities and four with no disability were aware of *Repeat Keys*, or knew that the key repeat delay could be altered. Nine of those with a disability and one other had chosen to alter the key repeat delay on their usual machine. All had either increased the delay or disabled repeats altogether.

Table 5.5 summarises, for each participant:

- Each participant's previous knowledge and use of the utility. Those who have altered the default key repeat delay on their usual computer are users, those who are aware of *Repeat Keys* but do not wish to alter the default repeat delay are not users, while those who were unaware that it was possible to alter the repeat delay are unaware.

- The key repeat delay they usually used (12, 16, 24, 40, off, or default). Where participants did not usually use a computer at all, the value 'n/a' is given. Where participants did not know what setting they normally used, a '.' is given in the table. Where participants used whatever the default was on the system they were using, the response 'default' is given.

Table 5.5: *Repeat Keys* preferences

Participant	Previous awareness	Usual delay (ticks)	Opinion before	Opinion after	% errors with default
ED1	user	24	very	.	6.4
ED2	user	off	useful	useful	4.7
ED3	user	24	somewhat	useful	19.0
ED4	unaware	24	very	essential	14.0
ED5	user	n/a	useful	.	4.9
ED6	unaware	default	not	.	2.1
ED7	unaware	default	not	.	0.0
ED8	not user	default	not	essential	21.0
ED9	unaware	n/a	don't know	don't know	2.1
ED10	unaware	off	useful	.	0.2
ED11	user	default	somewhat	no difference	1.1
ED12	unaware	default	useful	no difference	1.4
ED13	user	24	useful	useful	0.0
ED14	unaware	.	useful	.	1.2
ED15	unaware	default	useful	.	2.3
ED16	unaware	default	not	.	0.5
ED17	unaware	40	essential	essential	74.4
ED18	user	24	useful	useful	0.0
ED19	user[2]	24	not	not	0.0
ED20	user	24	useful	useful	0.6
EN21	unaware	.	somewhat	not	0.2
EN22	unaware	default	not	not	0.2
EN23	unaware	default	not	no difference	0.0
EN24	not user	default	not	not	0.0
EN25	user	40	useful	useful	0.0
EN26	not user	default	not	not	0.0
EN27	unaware	n/a	don't know	.	0.3
EN28	unaware	default	not	useful	0.0
EN29	not user	default	useful	no difference	0.0
EN30	unaware	default	not	somewhat	0.0

- Their opinion about *Repeat Keys* before using the facility (not useful, somewhat useful, useful, very useful, essential, don't know).
- Their opinion after having tried an altered repeat delay. Where participants did not try an altered delay, a '.' is shown in the table. Some participants, after having tried the facility, observed no difference from the default configuration.
- Their long key press error rate (% of deliberate alphanumeric keystrokes causing an error) before altering the key repeat delay.

[2] This participant used a 24 tick delay for the benefit of his non-disabled son, not for his own needs.

Before performing any typing, one participant viewed *Repeat Keys* as essential, two considered it very useful, eleven useful, three somewhat useful, eleven not useful, and two did not know whether it would be useful or not. These opinions were based on prior experience of *Repeat Keys* (14 cases), or a description of the utility (16 cases). After having tried using an altered key repeat delay, six participants changed their opinion. Five changed in favour of the utility, and one against. Of these six, four had no previous experience of the utility, while the remaining two usually used machines with a longer delay than the Macintosh default. No participant expressed any difficulty in using the facility once it had been activated for them. One reported that the keyboard itself felt lighter and more responsive. Four participants did not observe any difference when using a changed delay.

The Spearman Rho test was used to test for correlation between the participants' initial opinions and their error rates in the first passage typed. No significant correlation was found in either the disabled or non-disabled groups. However, the opinions of participants with disabilities after trying an altered repeat delay (excluding those who had observed no difference or did not have an opinion) did show significant correlation (Spearman Rho =-0.529, N=9, $p<0.05$) with these initial error rates.

Table 5.6 summarises the final opinions of those who used an altered delay, or had previous experience of *Repeat Keys*. Participants with and without motor disabilities are shown separately.

Table 5.6: Final opinions about *Repeat Keys*

	Essential	Very useful	Useful	Somewhat useful	Not useful	No difference	Don't know	Total
Those with a disability	3	1	6	0	1	2	1	14
Those with no disability	0	0	2	1	4	2	0	9
Total	3	1	8	1	5	4	1	23

Most participants with motor disabilities considered the facility at least useful, while most of those with no disability thought it not to be useful, or to make no difference.

5.7 Discussion - Repeat Keys

Under the default key repeat delay, some participants with motor disabilities had very high long key press error rates (up to 74.4%), and spent much time correcting these errors (up to 44.8%). Participants without disabilities showed low error rates (up to 0.3%). There was no correlation between error rate and experience in either group, which suggests that practice can reduce but not eliminate long key press errors. For sixteen of the thirty participants, fatigue or practice effects acted to increase or decrease key press lengths over the course of the experiment. All of those exhibiting fatigue had a disability affecting their typing. Those improving with practice included participants with and without disabilities.

The presence of these effects over a two hour typing period suggests that for many individuals, their ideal key repeat delay is not static. While users could avoid errors by setting the key repeat delay to the maximum they anticipate requiring, or by disabling repeats altogether, this slows or prevents the deliberate use of repeats, for example when using arrow keys for positioning. The effect of using long key repeat delays on word processing tasks has not been examined here. It has been assumed that the minimum delay which eliminates long key press errors is the optimal setting.

For those participants with non-zero error rates under the default delay who tried an increased delay, error rates were drastically reduced. Time spent correcting long key press errors in turn decreased. For those with the highest initial error rates, this represents a substantial saving in time and effort. No participant experienced difficulty in using the longer delay settings. Reductions in error rate were observed for all participants who tried an increased delay, despite four of this group exhibiting fatigue effects, and only two showing practice effects.

The use of *Repeat Keys* itself can also affect key press lengths. Participant ED3 made longer key presses when the delay was increased. This suggests that her natural, most comfortable key press length was often above the default, but that she could reduce her key press length to some extent to accommodate a shorter repeat delay when necessary. Some of the participants' own comments suggest that the keyboard was much easier to use with a longer delay. *Repeat Keys*, therefore, can not only reduce errors but also make the keyboard more comfortable to use.

Decreasing the key repeat delay below the default value was shown to introduce errors for both disabled and non-disabled participants.

In this small sample, the full range of available increased repeat delay settings were used. Even with the maximum available delay of 40 ticks, one user still showed long key press errors. This suggests that an increased range of settings, including settings greater than 40 ticks, would be beneficial for some users.

Half of the participants were previously aware of the utility, and one third had used it. It was primarily used by participants with disabilities, but one non-disabled user was also found. Of the sixteen previously unaware of the facility, twelve had non-zero error rates, and may have benefited from using it.

The participants' opinions were influenced by the active key repeat setting on the machine they normally used. In some cases, participants normally used a computer with a repeat delay longer than the Macintosh default, and were unaware that a problem could exist on alternative machines. Accurate knowledge of the repeat delay they used on their usual machine would have been helpful in interpreting their initial responses - most participants did not know the precise setting and the responses they gave were often guesses. Some may have been using an altered delay without being aware of it. Although these opinions showed a high level of enthusiasm about *Repeat Keys*, some methodological limitations of the study which may have influenced these findings are discussed in Section 5.10.

These results are encouraging, indicating that in the absence of any difficulty in activating and setting *Repeat Keys*, the facility can make an effective contribution to keyboard accessibility for computer users with motor disabilities. *Repeat Keys* was also considered useful by users who have no difficulty in adapting to the default repeat delay.

5.8 Results - Sticky Keys

Five of the thirty participants, all with a motor disability, typed with one hand only. A further seven, six with a motor disability, typed predominantly with one hand but could employ the other hand for modifier key presses. The remaining eighteen typed using both hands. Three of the two-handed typists, all with no disability, were novice touch typists.

5.8.1 Use of Modifier Keys

The data described in Chapter 4 suggests that a common input error due to difficulty in using modifier keys is the dropping error. Dropping errors occur when a user fails to maintain pressure on a modifier key, and it is released before the key to be modified has been pressed down. Several dropping errors may be made while typing a single modified character.

Not all users who have difficulty in holding down two keys at once make dropping errors. Some use techniques to avoid difficult modifier key presses. The most common technique is to use the *Caps Lock* key wherever possible.

Table 5.7 summarises the computer experience and typing style of the participants. For each participant, an error rate was calculated as the average number of dropping errors observed per use of the *Shift* key. This is shown in the fourth column of the table. The table also shows the number of times they used *Shift* while typing the first text passage, and whether they used *Caps Lock* to produce single capital letters.

The number of uses of *Shift* observed for each person varied between 11 and 38, the average being 28. 46 dropping errors were observed in the typing of the initial text passages, 45 of which were generated by participants with a motor disability. For each participant, an error rate was calculated as the average number of dropping errors observed per use of the *Shift* key. Five participants had error rates of 0.1 or greater. Participant ED6 made 31 dropping errors, giving an error rate of 1.0 - on average one error for every use of *Shift*. No other participant made more than three dropping errors.

In the majority of dropping errors observed in the first passage typed, the key to be modified was not pressed, no input was generated, and therefore no error correction was required. Only one participant spent time (22.7 seconds) correcting dropping errors in which a character was generated.

There was no significant correlation between experience and dropping error rate among the participants. There was, however, a significant inverse correlation between dropping error rate and typing style (Spearman Rho = -0.488, n=30, $p<0.01$). For the purposes of this analysis, typing style was measured on an ordinal scale of four points corresponding to one-handed, mainly one-handed, two handed and touch typing styles.

Table 5.7: Use of modifier keys

Participant	Experience (years)	Typing style	Dropping error rate	Uses of *Shift*	*Caps Lock* for single letters?
ED1	34.0	left hand only	0.13	16	yes
ED2	0.4	left hand only	0.04	23	yes
ED3	28.0	both hands	0.00	28	no
ED4	7.0	right hand only	0.18	11	yes
ED5	0.2	both hands	0.00	38	no
ED6	2.0	right hand only	1.00	31	yes
ED7	0.0	mainly right hand	0.00	33	no
ED8	0.0	both hands	0.00	24	no
ED9	0.1	both hands	0.06	31	no
ED10	5.0	both hands	0.00	28	no
ED11	28.0	both hands	0.00	26	no
ED12	0.2	both hands	0.00	13	yes
ED13	29.0	mainly right hand	0.11	28	no
ED14	0.04	mainly left hand	0.00	32	no
ED15	2.0	mainly left hand	0.00	32	no
ED16	1.0	left hand only	0.00	11	yes
ED17	6.0	both hands	0.10	31	no
ED18	7.0	mainly right hand	0.00	32	no
ED19	0.8	mainly right hand	0.04	26	no
ED20	12.0	both hands	0.00	29	no
EN21	2.4	both hands	0.00	30	no
EN22	9.0	touch typing	0.00	31	no
EN23	0.3	both hands	0.04	27	no
EN24	14.0	both hands	0.00	27	no
EN25	2.8	touch typing	0.00	34	no
EN26	18.0	both hands	0.00	30	no
EN27	0.0	mainly right hand	0.00	30	no
EN28	7.0	both hands	0.00	26	no
EN29	22.0	touch typing	0.00	27	no
EN30	0.0	both hands	0.00	27	no

All five one-handed typists used *Caps Lock* for single capital letters. Only one other participant - ED12 - used *Caps Lock* in this way. She was relatively inexperienced with the keyboard, and had not yet understood how to use *Shift* to produce capital letters. The use of *Caps Lock* to produce capital letters can introduce errors: on three occasions, a participant using this technique forgot to deactivate the lock. This occurred once each for Participants ED2, ED12 and ED16. These three participants spent a total of 16.3 seconds correcting this error.

5.8.2 Fatigue and Practice Effects

When using modifier keys, fatigue and practice could affect both the dropping error rate and the participants' ease of use of modifier keys. While practice may reduce dropping errors, the lack of observed correlation between previous keyboard experience and dropping error rate suggests that practice cannot eliminate the problem. Conversely, if fatigue were to increase users' difficulty in using modifier keys, and increase dropping error rates, the need for an alternative mechanism such as *Sticky Keys* would increase. During experimentation it was observed that some users spent time (and effort) manoeuvring themselves into an appropriate position before making simultaneous key presses. These efforts are highly likely to induce fatigue, but such effects are difficult to measure objectively.

Fatigue and practice could also affect the ease of use of *Sticky Keys*, and corresponding error rates. Because of the short period of time spent using *Sticky Keys*, these effects cannot be examined. As with *Repeat Keys*, fatigue and practice will act to degrade or improve performance. These effects are assumed to be present in some, but not all participants. The results presented in the following section are intended to give an overview of the use of the utility by a range of different people, in a range of stages of fatigue or experience. Negative transfer of learning effects due to the participants having already typed a passage without using *Sticky Keys* will be discussed in Section 5.8.5.

5.8.3 Using Sticky Keys

Both Section 5.8.1 and Chapter 4 have shown that people with motor disabilities, particularly one-handed typists, can have difficulty in using modifier keys. *Sticky Keys* offers an alternative mechanism for generating modified key presses, which is effective for all modifier keys, eliminates dropping errors, and does not require any movements other than single key presses.

This section examines the use of *Sticky Keys* by twenty-four of the thirty participants, including fifteen with some motor disability. Of the latter group, five typed with one hand only, six typed predominantly with one hand, but could use the other for modifier key presses, and four used both hands. Among the nine participants with no

disability, one typed mainly with one hand while the remainder, three of whom were novice touch typists, used both hands.

Some participants previously unaware of the utility commented that they found it "much easier" than the original method for generating modified characters. One predominantly one-handed typist described it as "more fluid", since he didn't have to stop to place his thumb on the *Shift* key. Some thought it would be easy to learn to use it, while others felt they were too used to the original method.

It is feasible that the use of *Sticky Keys* allows modified characters to be generated more quickly for some users. The time taken to produce modified key presses was measured as the total time between completion of the previous key press, and initiation of the following key press. For only one participant (ED15) were modified characters generated significantly faster when using *Sticky Keys* ($p < 0.05$, Mann Whitney U test). Furthermore, for eight participants, including three with a disability, modified characters were generated significantly more slowly when using *Sticky Keys* ($p < 0.05$, Mann Whitney U test). None of this group of eight were previously familiar with *Sticky Keys*. Two of them typed predominantly with one hand but could use the other hand for modifier key presses, the remaining six used both hands to type.

Dropping errors can no longer be generated when using *Sticky Keys*. In addition, none of the participants used the *Caps Lock* key to generate capital letters when *Sticky Keys* was activated, so no errors associated with this technique were observed. However, two new errors were introduced. The most frequent of these was unintentional deactivation of the facility by pressing a modifier key simultaneously with another key. When this happened, the experimenter reactivated *Sticky Keys* using the control panel. The second error occurred when participants pressed *Shift* more than once before pressing their chosen letter. Most often this occurred when a participant paused after having pressed *Shift*, forgot that they had already pressed it, and pressed it again. This activated the locking facility of *Sticky Keys*, causing future typing to be unexpectedly capitalised. When this occurred participants were given instruction on how to deactivate the lock.

Table 5.8 shows the participants divided into four groups according to their typing style: one-handed, mainly one-handed, two-handed, and touch typists. The participants' familiarity with *Sticky Keys* and errors made in using the facility are shown.

Table 5.8: Usage of *Sticky Keys*

Participant	Typing style	Familiarity	Accidental deactivations	Accidental locks
ED4	right	user	0	2
ED6	right	user	0	1
ED2	left	ex-user	0	0
ED16	left	unaware	0	0
ED1	left	occasional user	1	0
Average:			0.20	0.60
ED19	mainly right hand	user	0	0
ED14	mainly left hand	unaware	0	0
ED15	mainly left hand	unaware	0	0
EN27	mainly right hand	unaware	1	1
ED7	mainly right hand	unaware	2	0
ED13	mainly right hand	unaware	2	1
ED18	mainly right hand	not user	4	1
Average:			1.29	0.43
ED9	both	unaware	0	0
ED10	both	unaware	0	0
EN28	both	unaware	0	0
EN21	both	unaware	1	0
EN23	both	unaware	1	0
EN26	both	unaware	2	0
EN24	both	unaware	3	0
ED12	both	unaware	6	0
ED20	both	not user	1	0
Average:			1.56	0.00
EN29	touch	not user	2	0
EN22	touch	unaware	4	1
EN25	touch	unaware	4	0
Average:			3.33	0.33

Fourteen participants, including six with a motor disability, deactivated *Sticky Keys* accidentally a total of 34 times. Among the four groups, the greater the use of both hands, the larger the average number of accidental deactivations. On average, the touch typists made 3.33 accidental deactivations each, the two-handed typists made 1.56, the predominantly one-handed typists 1.29, and the one-handed typists 0.20. Only once did a one-handed typist deactivate *Sticky Keys*. The touch typists all deactivated it at least twice.

There was no significant correlation between the participants' previous experience of *Sticky Keys*, measured on a four point ordinal scale: 'never heard of it', 'tried and rejected it', 'sometimes use it' or 'always use it' and the number of times they deactivated it accidentally (Spearman Rho = -0.282). The experiment did not measure

the time that participants would have taken to discover that they had accidentally deactivated *Sticky Keys*, and to reactivate it.

Seven unintentional activations of the locking facility were observed while participants were using *Sticky Keys*. No participant reported noticing the visual feedback provided by *Sticky Keys*, which may have helped in avoiding errors of this kind. A total of 209.2 seconds was spent correcting errors of this type.

5.8.4 User Preferences

This section examines the actual usage of the *Sticky Keys* facility among the participants. Prior to the experiment, eleven of the participants with motor disabilities and one with no disability were aware of *Sticky Keys*. Six participants always or sometimes used *Sticky Keys*. All *Sticky Keys* users had a disability affecting their typing. Three typed with one hand only, one typed mainly with one hand, and two typed with both hands.

Table 5.9 summarises, for each participant:

- Their typing style: left or right hand only, mainly left or right hand, two handed, or touch typing.
- Each participant's previous knowledge and use of the utility. Those who always use *Sticky Keys* on their usual computer are users, those who sometimes use it are occasional users. Those who have tried the utility and used it in the past are ex-users, while those who tried it and never used it are not users. The remainder were unaware that the facility existed.
- Their opinion about the usefulness of *Sticky Keys* before trying the facility (not useful, somewhat useful, useful, very useful, essential, don't know).
- Their opinion after having used *Sticky Keys*. Where participants did not try the utility, a '.' is shown in the table.

Table 5.9: *Sticky Keys* preferences

Participant	Typing style	Previous awareness	Opinion before	Opinion after
ED1	left hand only	occasional user	somewhat	useful
ED2	left hand only	ex-user	useful	very useful
ED3	both hands	occasional user	somewhat	.
ED4	right hand only	user	essential	essential
ED5	both hands	user	very useful	.
ED6	right hand only	user	very useful	very useful
ED7	mainly right hand	unaware	useful	very useful
ED8	both hands	not user	not useful	.
ED9	both hands	unaware	don't know	don't know
ED10	both hands	unaware	useful	useful
ED11	both hands	not user	not useful	.
ED12	both hands	unaware	useful	very useful
ED13	mainly right hand	unaware	useful	useful
ED14	mainly left hand	unaware	don't know	very useful
ED15	mainly left hand	unaware	essential	essential
ED16	left hand only	unaware	don't know	very useful
ED17	both hands	unaware	useful	.
ED18	mainly right hand	not user	not useful	not useful
ED19	mainly right hand	user	useful	useful
ED20	both hands	not user	not useful	useful
EN21	both hands	unaware	not useful	not useful
EN22	touch typing	unaware	not useful	somewhat
EN23	both hands	unaware	very useful	useful
EN24	both hands	unaware	useful	somewhat
EN25	touch typing	unaware	not useful	useful
EN26	both hands	unaware	not useful	useful
EN27	mainly right hand	unaware	don't know	don't know
EN28	both hands	unaware	not useful	very useful
EN29	touch typing	not user	not useful	not useful
EN30	both hands	unaware	not useful	.

Before performing any typing, two participants viewed *Sticky Keys* as 'essential', three considered it 'very useful', eight 'useful', two 'somewhat useful', eleven 'not useful', and four did not know whether it would be useful or not. These opinions were based on prior experience of *Sticky Keys* (12 cases), or a description of the utility (18 cases). After having tried using *Sticky Keys*, thirteen participants changed their opinion. Eleven changed in favour of the utility, and two against. Of those thirteen whose opinion changed, ten had never previously used *Sticky Keys*, one occasionally used the utility, and two had tried and rejected it. Those three who had used *Sticky Keys* previously all revised their opinions in favour of the utility.

All of the five who typed solely with one hand described *Sticky Keys* as 'useful', 'very useful', or 'essential'. Of those six who typed mainly with one hand, four

considered *Sticky Keys* at least 'useful', one rated it 'not useful' and one was unsure. Those who typed with both hands gave answers ranging from 'don't know' up to 'very useful'.

Over the whole group, there was a significant correlation (Spearman Rho = 0.545, n=26, $p<0.01$) between the participants' error rate and their initial opinion of *Sticky Keys*. There was no significant correlation between error rate and final opinion (Spearman Rho = 0.249, n=22). The Spearman Rho test was also used to test for correlation between the participants' initial and final opinions and their typing style. A significant correlation was found between typing style and initial opinion (Spearman Rho = 0.548, n=26, $p<0.01$), and also between typing style and final opinion (Spearman Rho = 0.566, n=22, $p<0.01$).

Table 5.10 summarises the final opinions of those who used or had previous experience of *Sticky Keys*. Participants with and without motor disabilities are shown separately.

Table 5.10: Final opinions about *Sticky Keys*

	Essential	Very useful	Useful	Somewhat useful	Not useful	Don't know	Total
Those with a disability	2	7	5	1	3	1	19
Those with no disability	0	1	3	2	2	1	9
Total	2	8	8	3	5	2	28

Most participants with motor disabilities considered the facility at least useful. The opinions of those with no disability were divided: none thought it essential, but most thought they would find it of some use.

5.9 Discussion - Sticky Keys

Sticky Keys was found to be effective in tackling dropping errors and eliminating the need for awkward simultaneous key presses. However, many users found errors occurring through the use of the utility itself. The time participants spent recovering from these errors was greater than the time spent recovering from dropping errors and errors in the use of *Caps Lock*, despite assistance from the experimenter. While the

incidence of such errors was lower among the one-handed or predominantly one-handed typists, this may have been partly because the majority of the one-handed typists were already familiar with the utility. Furthermore, two of the two-handed typists were *Sticky Keys* users, so difficulties experienced by this group are worthy of attention.

Sticky Keys did not, in general, appear to significantly reduce the time taken to produce modified characters, in fact an increase in time was more often observed. The timing information on which these results are based also included times taken to read the next word and check previous work, and these, together with fatigue and learning effects, may have swamped any effects due to *Sticky Keys*. It is probable that the major benefit of *Sticky Keys* is in reducing the effort required to produce modified characters, rather than the time taken to type them.

Despite these negative observations, the utility was warmly received. It was considered useful by 79% of those who tried it, including six non-disabled participants. In general, participants' opinions correlated with their typing style - those using wholly or mainly one hand were more enthusiastic about *Sticky Keys*. However, many non-disabled, two handed typists reported that there were occasions when they did use the keyboard with one hand. *Sticky Keys* is useful for people with and without the disability for which it was originally intended to compensate.

It is notable that so many of the participants' opinions of *Sticky Keys* improved after having tried the utility. Description and demonstration of some utilities may not be enough to allow users to judge the utility's relevance to their requirements. This has implications for developers of access utilities and facilities which are intended to support configuration. An approach based entirely on questioning the user may exclude many potential users. However, the following section will discuss potential sources of bias in these findings.

The participants in this study were given only minimal training in the use of the utility, and sometimes stumbled upon more advanced features - the locking mechanism in particular. This lead to confusion, and could potentially cause users to reject the utility, if no support was available. For some participants it would be useful to be able to disable these advanced features. For others, training in the use of *Sticky Keys* may be helpful. Awareness of the utility was much higher among participants with disabilities, but there was one one-handed typist who was unaware of it. Unfortunately, the relevant Macintosh on-line documentation is difficult for a novice

user to find. In Windows 95, only some of the functionality is described in the on-line help documentation. The user is left to infer the available functionality from the options in the control panel, which provides only incomplete information. For example, the user can choose that two presses on *Shift* will cause it to lock, but no explanation is given as to how to deactivate the lock! The options available to users of configuration facilities should be more explicitly described in on-line help systems.

Participants all typed at least one passage using the default method for generating modified characters before trying *Sticky Keys*. This was a methodological limitation imposed by the larger study. The initial use of the default mechanism could have lead to transfer of learning effects, particularly for novice keyboard users, which may have influenced the number of accidental deactivations of *Sticky Keys* observed. However, this is actually a more general problem - the majority of novice keyboard users are likely to learn the default method first, and similar negative effects would be expected when transferring to *Sticky Keys*. In this respect, then, the experimental methodology may have mirrored the real world experience of many users. The automatic deactivation of *Sticky Keys* when users revert to the default method is intended as a form of automatic configuration. However, these results suggest that it may well work against users who are new to *Sticky Keys*, as it is easy for them to deactivate the utility without realising it, and may become confused when it suddenly fails to work. It would, perhaps, be an improvement if the utility was less easy to deactivate. One suggestion would be to allow automatic deactivation only after a long enough break in input. Alternatively, this aspect of *Sticky Keys* could be removed.

These potential difficulties for novice users of *Sticky Keys* may partially explain why the rate of uptake among those aware of *Sticky Keys* is lower than that for *Repeat Keys*. Certainly, the interface of the utility as a whole requires further empirical examination.

5.10 Methodological Difficulties

A number of features of the experimental methodology, as described in Section 5.3, are less than ideal, and may have affected some of the results presented here.

Firstly, it was not possible to control for, or accurately measure, order effects. The majority of subjects copied a passage using the default configuration, then one using

an altered key repeat delay, and then one other passage before finally trying *Sticky Keys*. Previous use of the default configuration produced some negative transfer effects in the following passages, increasing the number of errors. In some cases, the keyboard practice could also have reduced errors. An improved methodology might avoid these difficulties by using the default configuration both before and after the altered configuration, and allowing participants to practice with each configuration prior to data recording. In this scenario, only a single configuration facility could be investigated in a given session, otherwise fatigue effects may become marked in later passages typed.

A second difficulty with the methodology used stems from the fact that participants were aware that the aim of the experiment was to examine the usefulness of configuration facilities. In addition, questions about the facilities' utility were asked directly by the experimenter. Although the questions were initially asked in the same form, participants often required clarification of the question, introducing the possibility of bias in the way in which questions were explained. This difficulty was exacerbated in the several cases where the experimenter was known to the participant. The likely effect of these influences is that participants gave more positive assessments of the facilities than they might otherwise have done. Further investigation of these utilities is therefore necessary in order to validate the high level of enthusiasm reported here. A sounder methodology might eliminate direct questioning by the experimenter in favour of allowing participants to try out facilities and choose their own configuration.

5.11 Summary

Several aspects of the use of *Sticky Keys* and *Repeat Keys*, two of the best known and most popular keyboard configuration facilities intended to support keyboard users with motor disabilities, have been described. The facilities have been used by users with and without motor disabilities affecting their use of the keyboard, and have been found useful to both groups in reducing errors, irrespective of changes in typing characteristics due to fatigue or practice. The facilities have also been seen to increase user satisfaction for at least some users. These data complement the informal and anecdotal reports which predominate in the literature.

Long key press errors can cause considerable difficulties, and the *Repeat Keys* facility was an effective solution. *Sticky Keys* was similarly effective in eliminating dropping errors. It was not only used by one-handed typists, supporting observations by previous researchers (Glinert and York, 1992; Newell, Arnott, Cairns, Ricketts and Gregor, 1995; Vanderheiden, Lee and Scadden, 1987) that facilities designed to improve disability access are often more generally useful. While the altered key repeat delay provided by *Repeat Keys* proved easy to use, some potential difficulties with *Sticky Keys* have been observed and require further investigation, and possibly redesign.

Many participants, when asked to rate the usefulness of the facilities, gave different replies before and after using the facilities. Often the second reply was the more enthusiastic. This result, if validated by future studies, has implications for designers of facilities intended to support users in configuration, for example help systems. It suggests that providing direct experience of a utility strongly increases the likelihood that it will be used, in comparison to providing descriptions and demonstrations.

These two configuration facilities can potentially have a significant impact on keyboard accessibility, particularly for users with motor disabilities. In the case of *Sticky Keys*, however, the full potential of the mechanism is not currently realised.

CHAPTER 6

The Usage and Potential Effectiveness of Existing and Proposed Configuration Support Tools

In Chapter 2 a number of keyboard and mouse access facilities were described, all of which are intended to compensate for problems caused by physical difficulties in manipulating these devices. The data presented in Chapter 4 has provided a sample of detailed information about the nature and frequency of such problems, and can be used to examine the potential effectiveness of the existing facilities for the sample group. Chapter 5 has also provided empirical evidence of the effect of two keyboard configuration facilities on keyboard usability.

This chapter compares the keyboard errors observed with the facilities available. Mouse configuration facilities are also briefly discussed. Identification of specific classes of mouse error that are tackled by mouse configuration options is less easy than for keyboard errors, and fewer mouse configuration options are available. The focus of this chapter is therefore on keyboard configuration options.

The potential effectiveness of *Sticky Keys*, *Repeat Keys*, *Slow Keys* and *Bounce Keys* are discussed. The chapter goes on to examine the extent to which the existing keyboard facilities are used by people who could benefit from them, and discusses factors impeding their use. A new facility - *Overlap Keys* - is also introduced and discussed. Finally, methods for overcoming the observed barriers to use of these facilities are suggested.

6.1 Existing Keyboard Support

There is a great deal of anecdotal and informal evidence that keyboard access facilities can be extremely useful to some users with motor disabilities. For example, Millar and Nisbet (1993) observe that most physically disabled or clumsy users need the *Repeat Keys* feature, and that it can make the difference between being able to use the computer, and not being able to. They also report that *Sticky Keys* and *Slow Keys* are very basic requirements.

While the majority of long key press errors were suppressed in the study described in Chapter 4, the data indicate that many participants often press keys for longer than the default key repeat delay, in the absence of any requirement to press them more quickly. Chapter 5 has presented empirical evidence that some people have only a limited ability to adjust their key press lengths to comply with the default key repeat delay, while others can make adjustments but prefer to alter the repeat delay to allow a more comfortable key press length. The data described show that *Repeat Keys* is indeed effective in improving keyboard access in such cases.

A second important barrier to keyboard access was the requirement to press more than one key down at once. Twelve of the participants studied in Chapter 4 typed wholly or mainly with one hand. The *Sticky Keys* facility is specifically designed to support one-handed typists, and those who find it difficult to press two keys at once. Chapter 5 has presented empirical data on the use of *Sticky Keys*, and its effectiveness in eliminating problems. The study concluded that while it is effective in reducing errors, and in reducing the effort required to produce modified characters, there are some potentially serious problems with the current interface.

Unwanted key presses were a further source of errors found in the data. These can be subdivided into additional key errors and remote key errors. Both the keyguard and the *Slow Keys* utility are intended to help in the elimination of unwanted characters. Only one of the participants in the study described in Chapter 4 regularly uses a keyguard. This suggests that a software alternative to the keyguard may be useful.

The *Slow Keys* facility is described by Brown (1992) as being helpful for users who brush keys accidentally. He states that "By introducing a very small delay factor, the great majority of accidental keystrokes can be eliminated without significantly reducing typing speed" (Brown, 1992, p42). Table 6.1 shows, for the six participants of Chapter 4 most prone to additional key errors, the rate at which they made additional

Table 6.1: Projected effect of *Slow Keys* on additional key errors

Participant	Additional key error rate - % of total deliberate keystrokes	Longest additional key error (ticks)	Maximum delay to retain 90% of keystrokes without modification of typing style (ticks)	% of additional key errors eliminated	% of correct keystrokes longer than this value	Minimum delay to eliminate 90% of additional key errors (ticks)	% of correct keystrokes longer than this value
19	4.1	19	12	75	93	18	36
1	3.8	8	5	47	94	8	46
20	3.1	11	7	74	92	11	54
9	1.6	10	6	36	90	11	21
15	1.4	11	6	68	94	10	55
7	1.2	34	8	31	94	21	2

key errors, the length of their longest additional error key press, the maximum *Slow Keys* delay that would affect no more than 10% of their normal typing, the percentage of their errors this setting would eliminate, and the percentage of their deliberate keystrokes that are longer than the suggested delay value.

With these settings imposed, *Slow Keys* would in theory eliminate 31-75% of the additional key errors, while requiring users to deliberately lengthen the shortest 6-10% of their keystrokes. In the best case, 75% of the errors of the participant with the highest error rate could be eliminated by imposing a delay of 12 ticks. This would require the participant to lengthen only the shortest 7% of their deliberate key presses. This analysis assumes that the lengths of additional key presses are not affected by changes in the length of deliberate keystrokes. For errors caused by pressing two keys at once, longer deliberate keystrokes may actually incur longer additional key errors. Research is required to investigate the effect of *Slow Keys* on the lengths of additional key error keystrokes. In addition, it is not known how easily users can learn to deliberately lengthen keystrokes. Further examination of this aspect of the use of *Slow Keys* is also needed.

Table 6.1 also shows the delay that would be required in order to eliminate 90% of additional key errors. This would require users to deliberately lengthen 46-98% of their keystrokes. For the relatively low error rates observed in this study, *Slow Keys* may not be the best solution. *Slow Keys* can be useful for people with Parkinson's disease [P. Hawes, Foundation for Communication for the Disabled, personal communication], but none of the participants in this study used *Slow Keys*, and it is in general less frequently used than the other facilities discussed here.

Table 6.2: Projected effect of *Slow Keys* on remote key errors

Participant	Remote key error rate - % of total keystrokes	Longest remote key error (ticks)	Maximum delay to retain 90% of keystrokes without modification of typing style	% of remote key errors eliminated	% of correct keystrokes longer than this value	Minimum delay to eliminate 90% of remote key errors	% of correct keystrokes longer than this value
7	1.0	58	8	17	94	58	0
15	0.7	6	6	90	94	6	94
20	0.4	6	7	100	92	6	95
19	0.3	16	12	33	93	16	60

Table 6.2 summarises the projected effect of *Slow Keys* on remote key errors, using the same measures as Table 6.1. Remote key error rates observed were very low, and showed a large variation in lengths. Participant 7, whose remote errors were generally caused by leaning on a part of the keyboard without realising it, had long error keystroke lengths which could not be easily eliminated by *Slow Keys*. For Participant 20, a small delay could eliminate all remote key errors, but would require deliberate lengthening of 8% of her keystrokes. Since the original error rate of 0.4% is much less than this value, it is questionable whether the adjustment required would justify the saving in reduced errors. Again, research into the trade-offs in the use of *Slow Keys* to eliminate errors against the requirement to make longer keystrokes would be useful.

For one participant in particular, a high rate of bounce errors was observed. The *Bounce Keys* utility, available on all platforms except the Macintosh, is designed to handle such errors. The three participants most prone to bounce errors (Participants 6, 15 and 20) accounted for 75% of the 44 errors recorded. If a delay of 10 ticks had been imposed as the debounce time (minimum gap between two identical key presses), this would have eliminated 87.9% of these bounce errors. For these three participants, only 3 deliberate double keystrokes were separated by a gap of 10 ticks or less. This suggests that *Bounce Keys* could be very effective in reducing bounce errors, while retaining the great majority of deliberate double key presses, with no alteration of the user's typing style. If users were to deliberately pause between typing double letters, then all of the deliberate double letters could be retained, and a greater number of bounce errors suppressed. Further research is required to investigate the ease with which users can adapt their typing style in this way, and to examine the effectiveness of *Bounce Keys* in practice.

Missing key errors were also found to be a source of keyboard errors. No software access provision currently addresses missing key errors, since they cause no input to the computer. Transposition errors were observed in small numbers, and did not appear to pose a significant problem for the participants studied. No access facility tackles transposition errors.

6.2 Existing Mouse Support

In the study described in Chapter 4, difficulties were observed in almost every aspect of mouse usage - pointing, clicking, multiple clicking, dragging, and repositioning the mouse on the table. As shown in Chapter 2, short of replacing the mouse itself with an alternative pointing device or a keyboard alternative such as *Mouse Keys* (Lee and Vanderheiden, 1987), few software configuration options are available.

The ability to vary the timing of multiple clicks is often provided. In Chapter 4, it was observed that 14% of unsuccessful multiple click attempts were too slow to be recognised as such. When participants were asked to click the mouse as quickly as possible, five of those in the main group had average values at or above the default maximum used on the Macintosh architecture. This suggests that increasing the maximum allowable value could be useful, but research would be required to determine whether such an increase would also introduce new errors due to separate single clicks being interpreted as multiple clicks

Another common facility is the ability to alter the mouse tracking speed. Some participants reported that the fast tracking speed used in the experiment of Chapter 4 was more difficult to use than slower speeds with which they were more familiar. It is, however, difficult to predict the effect of altering the tracking speed from the data available, and a thorough empirical investigation would be required to establish effects on pointing speed and accuracy.

Users who have difficulty in dragging can often use a mouse but avoid dragging by using keyboard shortcuts, particularly for operations like text selection and as an alternative to the use of menus. Chizinsky (1990) describes keyboard access to the Apple Macintosh operating system, for example. The form and availability of such options often varies between different operating systems and applications.

Beyond these mouse configuration options, some software packages allow modification of the way the pointer moves for a given mouse movement. For example, the mouse movement can be restricted to horizontal or vertical motion, editing out any shakiness on the part of the user. Again, these are specific packages, and are not provided as part of the default operating system.

All of these options require further empirical research in order to identify how well, and for whom they improve access. There are few mouse configuration options widely available in operating systems. The remainder of this thesis focuses on keyboard configuration options, and the adequacy of existing mouse configuration options will not be discussed further here.

6.3 Usage of Keyboard Configuration Facilities

Another indicator of the success of the existing configuration facilities is the number of keyboard users who regularly use them. Among the twenty participants with disabilities described in Chapter 4, fourteen regularly used computers. Of these people, seven usually used a PC, five used a Macintosh, and two used a BBC Microcomputer. Table 6.3 shows, for each type of keyboard access facility, the number of participants regularly using computers for whom the facility may have been useful, the number of those people who had access to the facility, and the number who were using the facility. *Sticky Keys* was defined as potentially useful for all those who typed wholly or mainly with one hand, or whose dropping error rate was 1% or greater. *Repeat Keys* was defined as potentially useful for those participants for whom at least 1% of their recorded keystroke lengths were greater than 16 ticks. *Bounce Keys* was similarly defined as useful for those whose bounce error rate was 1% or over, and *Slow Keys* and the keyguard were considered potentially useful for those whose additional key error rate or remote key error rate was 1% or over.

All of the facilities were available to all the participants who might have benefited from them. *Sticky Keys and Repeat Keys* were used by half of those people. Three of the five participants who were not using *Sticky Keys* were unaware of its existence. The other two had tried it but had not found it useful. There were three participants who may have benefited from using a longer key repeat delay and did not use one. All three had access to a version of *Repeat Keys*, but none knew of its existence. No participant

Table 6.3: Usage of Keyboard Access Facilities

Facility	Number for whom it may be useful	Number with access	Number using the facility
Sticky Keys	10	10	5
Repeat Keys	6	6	3
Bounce Keys	0	0	0
Slow Keys	5	5	0
Keyguard	5	5	1

who regularly used computers had a bounce error rate over 1%. Five participants may have benefited from using a keyguard or *Slow Keys* to reduce additional key errors. One did usually use a keyguard. The remaining four participants had considered but rejected a keyguard because it was uncomfortable, slowed them down too much, or would interfere with their normal method of typing. None of them used *Slow Keys*, and it is not known whether they had tried it. Those participants who make many additional key errors often rely on spell checking programs to correct their work.

6.4 Barriers to Usage of Access Facilities

These results suggest that while some of the keyboard access functions available in standard operating systems are found to be useful and are actively made use of by their target population, others are less well used. In this study, the major barrier to their adoption was a lack of awareness of their existence.

The participants in this study were not asked what support was available to them in their use of the keyboard and mouse. In the study presented in Chapter 8, where twenty participants with motor disabilities were asked whether they had any computer support, of the eighteen who regularly used computers only seven had a teacher or tutor who could help them with configuration.

Lack of awareness, however, is not the only barrier. Two of the participants who were unaware of the access facilities used computers in an organisation which provided teachers who *were* aware of them. The teachers had deliberately not informed the participants of the facilities because the computers they used were shared by many people. The process of adjusting the computer before each session, and resetting it at the end was perceived by the organisation to be error-prone, time

consuming and not worth the effort. The two participants in question were not able to set and reset the facilities themselves. Another participant, when made aware of the existence of the access facilities, reported that he would not use them because the computer he used was not his own. At least eleven of the fourteen participants considered here use computers that are also regularly used by other people.

6.5 Overlap Keys - A Potential Alternative to Slow Keys

Section 6.1 discussed the extent to which effective use of *Slow Keys* to reduce additional and remote key errors would require users to slow down their typing. It also highlighted the possibility that longer deliberate keystrokes may have an effect on the length of additional key errors, which would lead to a reduction in the number of errors eliminated. Given the lack of empirical evidence describing the use of *Slow Keys*, it is not clear how effective or usable it really is for the majority of people who press keys accidentally.

Many of the additional key errors observed involved overlapping keystrokes. The majority of participants with disabilities, including five of the seven most prone to additional key errors, did not deliberately type overlapping keystrokes. An alternative to *Slow Keys*, focusing on overlapping keystrokes, might therefore be possible.

6.5.1 Overlap Keys

Overlap Keys is an experimental new keyboard configuration facility, implemented by the author. It is aimed at users who are prone to making additional key errors in which both the intended and unwanted characters register, and whose typing style produces very few deliberate keystrokes that overlap in time. One-handed typists, for example, tend not to deliberately overlap keystrokes.

Overlap Keys uses knowledge of the current keyboard layout to isolate instances of overlapping keystrokes on adjacent alphanumeric keys. There are a number of possible actions that could be taken on identification of such keystrokes. The simplest, and least helpful, is to eliminate both of the overlapping keystrokes. In terms of errors, this transforms a word with an extra, unwanted character, into a word with a

missing character. The evidence from the study of Chapter 4 suggests that users who make additional key errors are aware of these errors. For a slow typist who has made an error and has not already typed further characters, it may be easier for the error to be eliminated, so they can retry without having to delete the unwanted character (which may involve two actual deletion operations).

Another possibility would be for the utility to attempt to identify and eliminate only the unwanted character. The data presented in Chapter 4, suggests that timing information about the keystrokes may be helpful in distinguishing unwanted characters. For example, in 68% of cases, the desired character was the last to be raised. English digram/trigram information could also be used to help identify the correct character. In cases of uncertainty, the input could be left unchanged, or both characters could be eliminated.

As an alternative to the *Overlap Keys* utility, the information about overlapping adjacent keystrokes could be incorporated into a spell checking program, perhaps leading to faster, more accurate suggestions of replacements for unrecognised words.

6.5.2 Preliminary Investigation into the Feasibility of *Overlap Keys*

As a preliminary investigation into the potential of *Overlap Keys*, eighteen of the participants in the experiment described in Chapter 5 were asked, during the same session, to try a simple version of the *Overlap Keys* utility, in which overlapping keystrokes on adjacent keys were deleted. The utility gave no warning when deletions were made. Prior to using the facility, participants were given a demonstration of its operation. They were advised that both errors and deliberate keystrokes could be deleted. No practice session was given.

Table 6.4 shows the participants' opinions of the usefulness of the utility, after having been given a demonstration before using it, and after having used it. Responses were solicited on a six point scale: 'don't know', 'not useful', somewhat useful', 'useful', 'very useful', or 'essential'. In addition, after trying the utility a number of participants responded that they had not observed any difference when the utility was activated. These are listed as 'no difference' responses. The table also gives the participants' error rates (the number of additional key errors involving two or more overlapping key presses per correct character typed), and their tendency to deliberately

Table 6.4: Summary of use of *Overlap Keys*

Participant	Opinion before trying	Opinion after trying	Error rate in original passage	Error rate while using *Overlap Keys*	Original rate of deliberate overlaps	Rate of deliberate overlaps with *Overlap Keys*	Errors minus deliberate adjacent overlaps
ED18	useful	very useful	0.001	0.001	0.001	0.000	1
EN28	useful	useful	0.002	0.006	0.005	0.009	2
ED1	don't know	useful	0.013	0.006	0.011	0.003	4
EN24	useful	not useful	0.005	0.009	0.129	0.116	-31
EN25	not useful	not useful	0.009	0.007	0.127	0.130	-38
EN21	not useful	not useful	0.000	0.001	0.055	0.065	-11
ED20	somewhat	not useful	0.024	0.024	0.079	0.096	8
EN23	very useful	not useful	0.000	0.001	0.025	0.025	-13
ED10	useful	don't know	0.002	0.002	0.002	0.003	1
EN22	useful	no difference	0.000	0.000	0.038	0.030	-3
ED3	don't know	no difference	0.007	0.005	0.044	0.046	-9
EN26	useful	no difference	0.004	0.009	0.030	0.010	5
ED11	useful	no difference	0.004	0.005	0.167	0.161	-16
ED5	not useful	no difference	0.013	0.013	0.013	0.015	10
ED6	somewhat	no difference	0.032	0.042	0.035	0.037	44
ED7	somewhat	no difference	0.007	0.011	0.006	0.011	7
ED13	very useful	no difference	0.005	0.008	0.003	0.006	5
ED2	useful	no difference	0.005	0.001	0.004	0.001	1

overlap keystrokes (the number of overlaps per correct character typed, excluding modifier keys). The difference between the number of additional key errors and the number of deliberate overlapping key presses on adjacent keys while *Overlap Keys* was being used is given in the final column of Table 6.4. A positive number means that most of the character pairs analysed by the utility were errors, while a negative number means that most were deliberate overlapping keystrokes.

Thirteen of the eighteen participants responded positively to a description and demonstration of the utility, but only three thought it useful after having tried it. The majority reported that they had not noticed any difference when the utility was activated, while three concluded it was not useful. Among those eight who were aware of the operation of the utility, there was a significant inverse correlation (Spearman Rho = -0.87, $p<0.01$) between the overlap rate of the participant, and their enthusiasm for the utility. The more often a participant naturally overlapped keystrokes, the more likely they were to have rejected the utility. There was no significant correlation between error rate and final opinion for this group (Spearman Rho = -0.34 for the error rate when *Overlap Keys* was used). This may be due to the

low error rates observed among this sample, and the strong effect of overlap rate. Among the nine participants with overlap rates less than 0.01%, the three participants who did notice the utility's operation (Participants ED1, ED18 and EN28) all considered it useful or very useful.

The difference between number of errors made and number of deliberate overlapping keystrokes on adjacent keys, given in the final column of Table 6.4, is dependent on the four text passages used, and on the typing styles of the participants. For only one participant (Participant ED6) was the number of errors markedly greater than the number of deliberate overlaps. It seems that the majority of the participant group did not fall into the class of typists for whom *Overlap Keys* might be useful.

Nevertheless, Participant ED11, who did not notice the operation of the utility and had a high rate of deliberate overlaps, thought he would have used the facility if it was attached to a dictionary and was reasonably good at guessing the correct letter. Participant ED20, who rated the utility as 'not useful' and had a fairly high natural overlap rate reported that she would use such a facility if it guessed the erroneous character and deleted it, and was at least 50% accurate. Participant EN28 felt that a beep to warn when overlapping characters were recognised would be helpful.

Given that the simple version of *Overlap Keys* used in the experiment deleted at least one pair of deliberate keystrokes for the majority of participants, it is surprising that the error correcting behaviour of the utility was so little noticed. A version of the utility with a more constructive error correction mechanism which attempted to guess the intended letter would produce fewer errors in the final text, and presumably be even less noticeable to users. It would be worthwhile developing a more advanced version of the utility, and performing evaluation on a more appropriate participant group, with lower deliberate overlap rates and higher additional key error rates than those described here.

6.6 Suggested Mechanisms for Increasing the Usage of Keyboard Configuration Facilities

The analysis presented in this chapter has illustrated that, while the right keyboard configuration facility can be extremely helpful for the right person, a number of problems currently exist:

- Some commonly observed difficulties presented in Chapter 4 are not addressed by existing software facilities. Many additional key errors and remote key errors may not be tackled effectively by *Slow Keys*, for example. There are opportunities for exploration of new utilities such as the prototype of *Overlap Keys* presented here.

- For at least one utility, aspects of the interface design could pose a serious barrier to usability, particularly for naïve users. Thorough usability studies are recommended for all of the utilities, in order to identify potential difficulties. The automatic deactivation of *Sticky Keys* the first time the user does not use the alternative method for generating modified characters is one example. There is certainly scope for improvement in the interface presented to the novice user in this case.

- Users may not be able to discover or operate access features themselves, particularly when new to computing. It is generally not obvious that such features exist, or how to activate them. While availability within modern operating systems is good, and packages providing the facilities are available for many older systems, many users remain unaware of these facilities. One way of surmounting this problem may be to provide more active on-line support for configuration. For example, Microsoft are currently developing an 'Accessibility Wizard', intended to help people to configure their machines by describing the facilities available, asking the user about their requirements, and implementing the user's chosen configuration. The results presented here suggest that active support should also include encouraging users to try the utilities.

Active configuration has a number of attractions beyond simply making users aware of facilities they may be interested in. It may also present a solution to the difficulties observed where computers are shared by many different users, given the perceived difficulty of adjusting the access settings for each user. An active configuration utility could help to ease the transition between users, allowing them to configure shared machines without the help of a teacher or system administrator.

Operating systems such as Windows 95 do provide a multi-user facility which allows settings for individuals to be stored and recalled. While this would alleviate the problem in situations where a static set of users could be identified, it does not help on machines open to the general public.

As more access facilities are developed, the large range of options available will exacerbate the problem of informing users of facilities they may wish to use. Ideally, a configuration support tool should identify a subset of the available facilities that are relevant to the current user, and focus the interaction on that set. A mechanism to diagnose the current user's requirements would also be useful for identifying when the current configuration was no longer appropriate - the original user may have left, or the user's needs may have changed during a session due to fatigue or practice.

The following chapter describes the development and implementation of a model of keyboard skills, designed to recognise keyboard performance errors and choose appropriate configuration facilities, and settings for those facilities. Given such a model, dynamic configuration support of the kind envisaged here becomes feasible.

Chapter 7

A Model of Keyboard Configuration Requirements

As described in Chapter 3, user modelling and adaptive interfaces have much to offer the field of input device configuration, given the existing barriers to configuration discussed in Chapters 2, 5 and 6, particularly the lack of knowledge about facilities and how to use them. This chapter presents a model of keyboard skills and configuration requirements, following the requirements outlined in Chapter 3.

Firstly, in Section 7.1, the focus on keyboard facilities at the expense of mouse facilities will be explained and justified. In Section 7.2, the requirements of the problem domain are related to the established user modelling techniques that were introduced in Chapter 3. Section 7.3 gives an overview of the architecture of the model, while Section 7.4 provides a detailed technical description of the algorithms used. Initial evaluation of the model, and tuning of model parameters, were carried out using the recorded typing data described in Chapter 4. This evaluation, and its results, are summarised in Sections 7.5 and 7.6.

A description of the model, and the results of this formative evaluation, have been published in Trewin and Pain (1997).

7.1 Domain to be Modelled

While data describing both keyboard and mouse usage are available, mouse difficulties and their solutions are not tackled by the model. Annotation of the original log files showed that even in the presence of detailed task knowledge a user's mouse actions could be very difficult to interpret by hand. For example, sometimes only the video

evidence was sufficient to show the user's reason for abandoning an attempt to drag, or to differentiate between accidental clicks and deliberate ones. Keyboard errors proved easier to define and identify. In addition, a much greater range of configuration options exists for keyboard errors than mouse ones, as illustrated in Chapter 2. Identification of keyboard difficulties therefore offers a more immediate reward in terms of improving usability of keyboards. Very often, the best solution available for mouse difficulties is to avoid using the mouse altogether.

As a starting point for investigation of automatic configuration, keyboard difficulties therefore presented the more promising area for initial research. The model described in this thesis has been further restricted to cover only those keyboard performance errors which occurred relatively frequently in the study described in Chapter 4, and which are also addressed by existing configuration facilities. There are four such errors: long key press errors, difficulty in using modifier keys, additional key errors in which two or more keys were activated, and bounce errors.

The characteristics of keyboard and mouse data are very different, and it is not clear that a single approach would be appropriate for both. A discussion of potential approaches to automatic detection of mouse difficulties can be found in Chapter 9.

It is envisaged that the model of keyboard skills would be most useful on shared or public access machines with a large number of different users, including novice users. In general, the configuration requirements of different users with motor disabilities vary enormously. Individual variation is also possible, as illustrated by Participant 19, described in Chapter 4, who reported using different configurations, according to the platform she was using and how well she was feeling. For example, on a bad day she uses *Sticky Keys* on her Macintosh Powerbook, on a good day she doesn't. This variation could occur between sessions, as in the case of Participant 19, or within a session, due to factors such as fatigue, or improvement through practice.

In multi-user situations, there may be no explicit indication of changes in user, and the previous user's configuration may conflict with that of the next user. The data reported in Chapter 6 suggest that many users of shared machines are reluctant to perform configuration that is not required by other users of the machine. Dynamic configuration support, guided by a user model such as this, has the potential to eliminate difficulties caused by the conflicting requirements of different users, while ensuring that each individual works within a suitable environment.

The problem of choosing an appropriate keyboard configuration is in many respects a typical user modelling problem. A mechanism is required to identify configuration options relevant to the current user, and settings for those options. The motivation behind the model is to have the system take greater responsibility for ensuring that the user can communicate successfully with it.

In other respects, this is an atypical user modelling problem. The majority of user models are concerned with cognitive aspects of users: their knowledge, interests, skills, preferences, etc. These cognitive qualities are to be inferred from the physical inputs provided by the user, which may contain inconsistencies, typing mistakes, or a host of other factors obscuring the user's actual cognitive state. In addition, users change over time, and these changes must also be handled. User modelling is, therefore, a challenging field of research. The domain addressed by this thesis is, in many ways, less complex than typical user modelling problems: physical inputs are used to infer physical, rather than cognitive, information about the user. Nevertheless, the keyboard skill model is required to handle uncertainty in interpretation of users' inputs, inconsistency in their manifestation of difficulties, and changes in their configuration requirements over time. The model is based on a data source that allows direct measurement of users' input behaviour, and no error-prone inference of higher-level qualities from physical input is required.

7.2 Relation to Existing User Modelling Techniques

In Chapter 3, Section 3.3 a number of desirable characteristics of a model of keyboard configuration requirements were identified. It should be:

1. *Individual*: tailored to the current keyboard user.
2. *Implicit*: based on interpretation of the user's normal typing rather than explicit questioning or testing. This approach allows large volumes of data to be examined, while enabling the model to draw conclusions quickly from a small sample of keystrokes.
3. *Dynamic*: updated continuously throughout a session. The model must be sensitive to medium term variations in the user's typing characteristics, so that the configuration can be altered as the user's requirements change. The model should

also adapt quickly to sudden changes of user. Because of the uncertainty in the interpretation of an input stream, the model must also be tolerant of errors in the performance error recognition mechanisms.

4. *Short term*: using only recent typing data. While long term (between session) characteristics could be usefully included, this would restrict the model to situations where users could be reliably identified, in order to facilitate recall of stored models.

Despite the problem of interest being characterised as a traditional user modelling problem, many of the common techniques used are not suitable here. Those that rely on stereotyping (Rich, 1989) are not applicable. While stereotypes based on disability could use knowledge that a user had lost an arm as a result of an industrial accident to infer that they may be interested in using *Sticky Keys*, knowing that the user's disability was the result of a stroke would provide little useful information, due to the enormous individual variation in the after effects of a stroke. The data presented in Chapter 4 suggests that similar keyboard problems may stem from very different disabilities, and similar disabilities may produce very different performance errors. For example, Participant 1 and Participant 10 had both had strokes. While Participant 1 made many additional key errors (73) and missing key errors (17) but no long key press errors, Participant 10 had an extremely high long key press error rate (377 in a single text passage), but made very few additional (3) or missing key (0) errors. This data certainly does not constitute a basis on which we could begin to define useful disability-based stereotypes, should they exist.

It is similarly difficult to draw inferences from observation of one aspect of typing (e.g. key press length), about a user's ability in another aspect (e.g. use of modifier keys), since the attributes studied are not strongly related. The data available provide no evidence that such inference is valid, so stereotypes based on relationships between areas of difficulty are also not feasible, given the existing data.

Overlay models (Clancey, 1987), simply identify areas where a student lacks the knowledge of an expert. In order to model keyboard difficulties and make configuration recommendations it is necessary to represent not only the skills the users have, but also the skills they lack, and the way in which their typing differs from that of an expert. In this domain, the goal is not to enable the user to become an expert, by identifying problem areas and misconceptions, but to actively accommodate those problems by recommending methods for alleviating or eliminating them. It is,

therefore, not enough to know that a user has difficulty in removing their finger from a key before the auto repeat facility engages. In order to decide an appropriate setting for *Repeat Keys*, information about the timing of the key presses is also necessary.

Approaches using bug libraries (e.g. Brown and Burton, 1978) are also either too restrictive or inappropriate. Such models capture information about how a student's skills and knowledge differ from those of an expert, and are dependent on knowing the user's task, and either identifying missing knowledge or hypothesising about the reason for any incorrect answers. Since free text input is desired, a mechanism reliant on knowing the text a user was trying to type would be too restrictive. A further constraint on such approaches is their assumption of consistency in the user's behaviour. Keyboard errors are highly inconsistent, in that they do not occur at every possible opportunity: not every key press will be too long, for example. It is the frequency of errors that indicates those with genuine difficulties. Even in teaching domains, for example the arithmetic domain investigated by Brown and Burton (1978), this assumption can cause problems (Self, 1988). Any technique for modelling keyboard skills must deal in frequencies, rather than binary values like known/not known. The high noise levels anticipated in the input of users with physical disabilities also make machine learning techniques inappropriate for this application.

The model should also be capable of managing uncertainty over the classification of a character sequence as being correct or containing some performance error. Uncertainty arises here because the user's task is unknown. Established numerical techniques for managing uncertainty - Bayesian networks and Dempster-Schafer theory (Jameson, 1996) - are not ideal. Bayesian networks could in principle be applied, but the full power of this technique is not required, due to the small number of sources of evidence available. Similarly, the ability of Dempster-Schafer theory to combine pieces of uncertain evidence is also not required, as the information sources available are reliable. While a single event (e.g. raising the *Shift* key immediately after having pressed it) could have a number of alternative interpretations (in this example, the event could be a dropping error, the user could have pressed the key accidentally, or the user could have deliberately released the key), it is only the probability that the event was a dropping error that is of interest here, rather than the best interpretation of the event. Neither does the problem require the handling of input involving vague or uncertain concepts, so fuzzy logic (Zadeh, 1994) is also inappropriate.

Given the simplicity of the data available, less complex (and less theoretically motivated) criteria have proved adequate for decision-making. The model uses simple domain-specific statistical techniques in order to recognise difficulties and handle uncertainty associated with the recognition process. While no knowledge of the user's task is required, an assumption is made that they are typing English text. Other languages, including command and programming languages, could easily replace or extend the model's knowledge about English.

The following section provides an overview of the functionality of the model. This is followed by a technical description of the model design and implementation.

7.3 Model Overview

In the following description, the term 'user model' refers to the code which implements the analysis of the current user, including the data structures summarising the recommended configuration.

The model of typing abilities focuses on the four classes of performance error for which some compensatory mechanism exists or has been proposed: long key presses, use of modifier keys, additional key errors and bounce errors. Analysis of these areas is carried out unobtrusively by trapping and examining keyboard events before they are passed on to the application in use.[1]

The structure of the user model is outlined in

Figure 7.1. There are three modules: input, event interpretation, and output of results. Event interpretation is implemented by five functions. The first manages the data structures representing current and recent events, while the remaining four examine the four aspects of typing that are of interest. User-specific data stored includes both general information about the user's typing characteristics and specific information about the recommended keyboard configuration for the current user. These data structures are dynamically updated as evidence about the current user's typing abilities is gathered. Threshold values and decay of evidence over time are used to damp out the effect of small variations in typing style, and of uncertainty associated with the recognition of specific difficulties. No changes are made to the actual keyboard

[1] The software used to trap input events will be described in detail in Section 7.4.1.

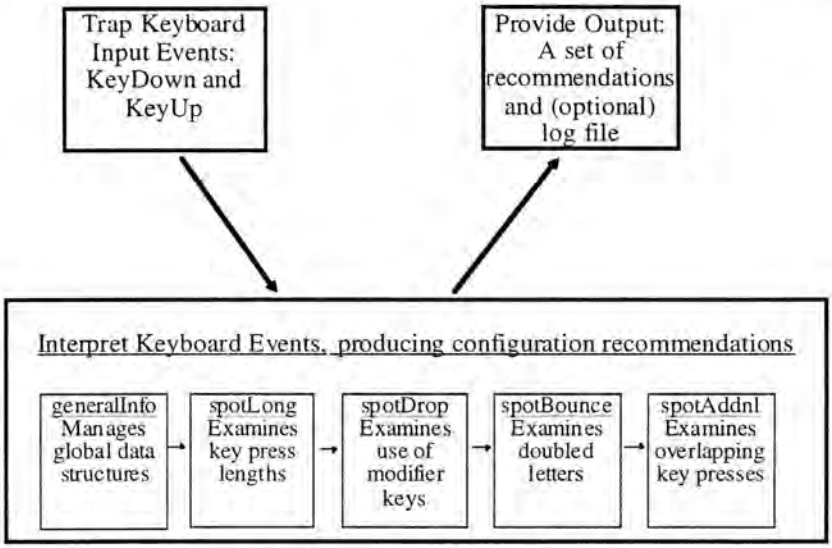


Figure 7.1: Overview of the model structure

configuration in use - the model simply makes recommendations. The control panel which comprises the model interface is shown in Figure 7.2. Modelling and logging can be switched on and off using the buttons provided, and the current evidence values and recommended settings for the access facilities are displayed and dynamically updated.

7.3.1 Choosing a Key Repeat Delay Setting

The key repeat delay chosen for the current user is based on their average key press length, and the amount by which their key presses tend to vary upwards from that average value. These calculations are limited to alphanumeric keys. In addition, abnormally long key presses are ignored, on the basis that they are likely to be deliberate, or caused by an event such as the user leaning on the keyboard. The recogniser chooses a value which is longer than at least 98% of the key presses considered.

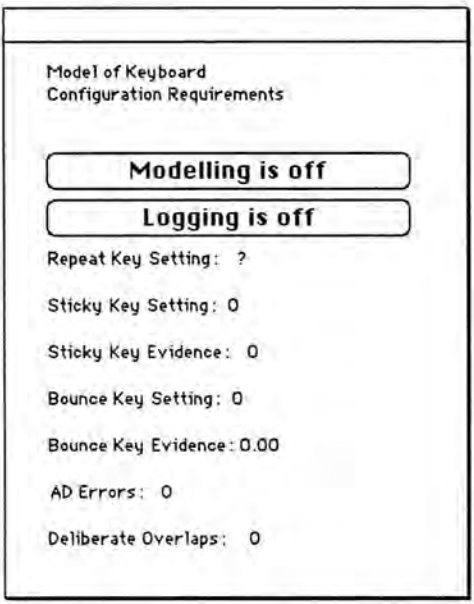


Figure 7.2: The control panel interface of the model

7.3.2 Assessing the Need for Sticky Keys

Assessment of the use of modifier keys is based on the observation that, in the data presented in Chapter 4, participants who had difficulty in pressing two keys at once would often type characteristic keystroke sequences, or adopt specific strategies for avoiding multiple key presses. Recognition of difficulties in pressing more than one key at once is based on the detection of such patterns, and these patterns are weighted according to the strength of the evidence they provide. Indicative patterns include:

- Use of *Caps Lock* for a single character.
- Pressing of a modifier key, followed by a small letter, followed by the *Backspace* key.
- Starting a sentence with a small letter.

Evidence from the observation of such sequences is accumulated, and the total value decays as the number of uses of modifier keys increases. Threshold values are used to decide whether *Sticky Keys* should be recommended on the basis of the current level of evidence.

7.3.3 Recognising Additional Key Errors

In 95.5% of the additional key presses observed in which the intended key was activated, the unintended key press overlapped in time with that of the intended key. Given this observation, all overlapping keystrokes are candidate additional key errors.

Since the user is assumed to be typing English, the model stores information about English digram frequencies. A digram is a pair of characters, and its frequency is the percentage of times that the second character immediately follows an appearance of the first character. Using knowledge of the keyboard layout, digram frequencies, and the current user's typing style, each overlap is classified as deliberate, an error, or of unknown cause. In the data available, 77% of the participants rarely or never deliberately overlapped keystrokes, so the user's typing style is an important source of information in this process.

The model counts deliberate and erroneous overlapping key presses. This information could then be used to assess the suitability of keyguards, *Slow Keys*, or *Overlap Keys* for a given user. The model itself could be extended to make such recommendations. This would require further research in order to establish appropriate threshold levels.

7.3.4 Assessing Bounce Keys Requirements

Detection of bounce errors is the most difficult of the four areas tackled by the model. Many people who make bounce errors are also capable of fast deliberate double key presses. The recogniser therefore has two challenges: to spot people who are making bounce errors, and to select a delay which will minimise the effect on the typing of deliberate double letters, while eliminating as many bounce errors as possible.

The recogniser operates by examining all double letters and assessing their likelihood of being bounce errors. Knowledge of the current user's previous typing, digram frequencies and the timing of the current double letter is used. For each double, an

evidence value between zero and ten is calculated. The greater the value, the higher the system's confidence that a bounce error has occurred.

The choice of value for the delay to be imposed is conservative, preferring to miss some bounce errors, rather than eliminate deliberate double key presses or require the user to alter their typing style.

7.4 Technical Description of the Model

The model outlined above has been implemented in ANSI C using the Metrowerks® CodeWarrior™ development system (Metrowerks Inc., 1993) on a Power Macintosh 6100/66, and runs on any Macintosh machine which uses or can emulate 68K processor assembly code. The structure of the model is illustrated in

Figure 7.1, which shows three major modules: input, processing, and output. Each of these are discussed separately below.

The application as a whole is a system extension/control panel combination. The system extension traps keyboard events, while the control panel builds up a model of the current typist. It can also produce ASCII log files similar to those of *InputLogger*, charting the input events and the conclusions drawn by the model about those events, to aid evaluation of the model's accuracy. The format of these log files, and an example log file, are shown in Appendix B.3. Modelling and logging are independently switched on and off via the control panel. When active, the model processes events regardless of what application is in use, and users may switch between applications as often as desired.

The model is fast and unobtrusive, being implemented at a low level in the system software. The use of separate processes for trapping input and processing events allows keystrokes to be stored up during busy periods, and examined during a pause. This helps to minimise the effect of the model on the response time of the running application. In practice, use of the model had no visible effect on the response of the word processing application used in the final evaluation described in Chapter 8.

7.4.1 Input: Keystroke Trapping on a Macintosh

The Macintosh operating system generates three types of keyboard event:

- `KeyDown`: an alphanumeric key has been pressed down.
- `KeyUp`: an alphanumeric key has been raised.
- `KeyRepeat`: an alphanumeric key is being held down, generate another instance of the appropriate character.

`KeyUp` events are often suppressed or ignored, as the majority of applications do not require them.

A time and key code are associated with each event. The time is measured in ticks, and is presented as the number of ticks since system start-up. The key code identifies the position of the key on the keyboard, and allows the system to choose the appropriate character to be generated, according to the keyboard type from which the event originated.

Keyboard events are trapped by the system extension software, which is a patch on the `SystemEvent` internal Macintosh routine. The control panel provides an on-off mechanism, processes the events, and also examines modifier key presses.

The control panel and system extension communicate using Apple's Audit© library. This provides an unobtrusive event tracing facility, usable from all types of Macintosh code segment. Events can be written into a store and read from the store. Events are written by the system extension, and read by the control panel. The model can set the size of Audit's internal store and the behaviour of the library when the store becomes full. If the internal store is full, then the oldest event in the store is overwritten. Audit warns whenever events have been lost, and the model reports an error if this happens. In practice, no events were lost during evaluation of the model, as the internal store was large enough (64 entries) to accommodate the speed at which users typed. The same value was used by *InputLogger*, in which it proved adequate for recording typing rates of 150 words per minute (Trewin, 1998).

Trapping alphanumeric keystrokes and mouse events

When the model is activated, the control panel initialises the Audit record. While the Audit record is active, the system extension examines all events reported to the `SystemEvent` routine. Whenever an alphanumeric key is pressed, a `KeyDown` event is generated. The system extension puts the event, including the time at which it happened, and the associated key code, into the Audit record. The control panel reads the event from Audit, along with a count of how many events, if any, have been lost, and passes this information into the model. When the key is released, a `KeyUp` event occurs and is treated in the same way. `KeyRepeat` events are ignored, as they are not required by the model.

Trapping modifier key presses

Pressing of modifier keys such as *Shift* or *Command* does not cause any keyboard events. Instead, internal flags are set. These modify the key code reported for other key presses, so that each alphanumeric key is modified in different ways by the flag settings in place when it was pressed down. This treatment of modifier keys makes it difficult to record accurate information about when the keys are pressed.

In order to measure the use of modifier keys, the system extension checks the status of all modifier keys every time an event is reported to the system extension. When a change is detected the appropriate `KeyDown` or `KeyUp` event is lodged with Audit.

When no events are occurring, the Macintosh generates NULL events. These are not reported to the system extension, but the control panel has access to them. In order to be able to record the usage of modifier keys as accurately as possible, the control panel examines their status on every NULL event. This mechanism results in occasional duplication of information recorded independently by the system extension and control panel. The user model simply ignores duplicated modifier key events. While this trapping mechanism is less accurate than recording of alphanumeric key presses, it is the best that can be achieved on the Macintosh architecture.

There is one key that cannot be recorded in the same way as the others: the *Caps Lock* key. This key is similar to other modifier keys in that no events are generated when it is pressed. Instead, its status is recorded in internal structures and can be read. However, because it is a locking key, when it is pressed the appropriate status bit is set

and this bit remains set until the key is pressed again. The length of time for which the key was actually pressed down each time cannot be established. The software reports a single `KeyDown` or `KeyUp` event for each press of the key.

Potential Extension Conflicts

The use of Macintosh system extensions, while providing a very efficient logging mechanism, also introduces the potential for conflicts between the model and other system extensions. Any extension using the Gestalt selector 'kmod' will clash with the model. Similarly, any program writing to a file with the same path name as the log file may also clash. No other specific clashes are known, tail patching is not used, and other instances of the Audit library will not cause problems. A Gestalt selector or file name conflict can be avoided by changing the Gestalt selector or file name used by the model, recompiling and reinstalling, or by disabling the extension which clashes.

7.4.2 Analysis: Interpreting Input Events

The core of the program is the central module which builds a user assessment from incoming keystrokes. This section describes how the model interprets these keystrokes, and how it incorporates knowledge of the keyboard layout and the English language. As illustrated in

Figure 7.1, the code consists of five main functions. The first is concerned with managing the event stream, and the remaining four handle the four aspects of typing examined by the model. Each module is described separately below, followed by a description of the data structures representing keyboard layouts and digram frequencies.

Managing the Event Stream

The `generalInfo` routine manages a data structure which keeps an up-to-date record of which keys are currently pressed down, and which keys have recently been released. It matches incoming `KeyUp` events with previous `KeyDown` events, taking

into account differences in the key codes caused by changes in the status of modifier keys between the two events. It ignores repeated modifier key events, as these are to be expected due to the nature of the trapping mechanism on the Macintosh.

The function calculates and stores information about key press lengths, gaps between key presses, overlapping key presses and double letters, all of which is used by the other modules. Storing gaps between key presses and double letters creates dependencies between keys, which has the effect that even when a keystroke is completed, it may still need to be stored. A maximum of 15 key presses can be 'active' at one time. An active key press is one which is unfinished, or is finished but some other key press which overlaps or immediately follows it is unfinished.

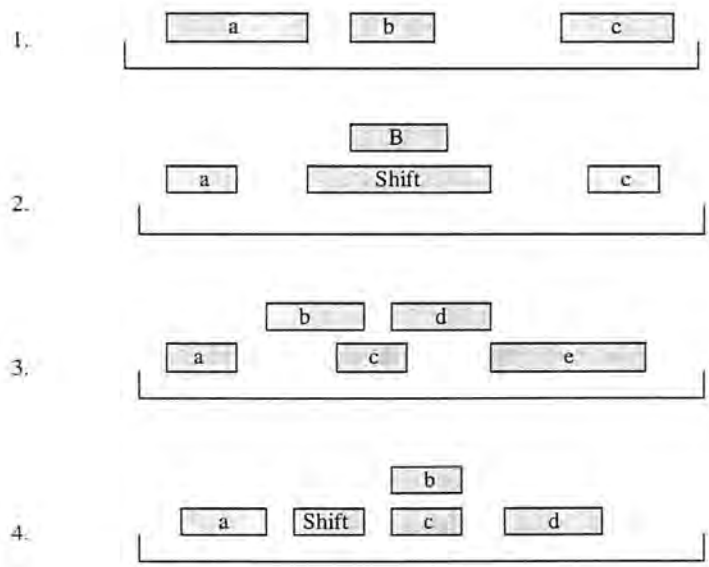


Figure 7.3: Some example keystroke timing patterns

Measurement of the gaps between key presses is simple when only one key is pressed down at a time. However, it is common for keystrokes to overlap. Figure 7.3 shows some example input streams. Part (1) illustrates the most basic case, where the gap between keys 'a' and 'b' is simply the time between the KeyUp event of 'a' and the KeyDown event of 'b'. In general, when a KeyDown event occurs, the preceding key is defined by the last keyboard event that occurred, regardless of whether this was a KeyDown or a KeyUp event. The gap between the key presses is calculated as the time between the end of the key pressed down first and the beginning of the second key press. Under this definition, when keystrokes overlap, as in case (2), the second key to be pressed down ('B') will have a negative value as the gap to the previous key ('*Shift*'). In case (3), 'b' is the preceding key for both 'c' and 'd'. For 'e', where the 'd' was raised at exactly the same time as 'e' was pressed down, the preceding key chosen is dependent on the order in which the KeyUp of 'd' and KeyDown of 'e' events are processed. In case (4), it does not matter whether 'b' or 'c' is chosen as the preceding key for 'd' - the value calculated will be the same.

Checking for double letters occurs when a KeyDown event is reported. All the active KeyUp events are considered as candidates for a previous matching key press. The last KeyUp event is always active. Other KeyUp events are active if they occurred after the last KeyDown event, or if their previous key, as defined above, is still pressed down . In case (1), for 'b', only 'a' is examined. In case (2), for 'c', both '*Shift*' and 'B' are examined. In case (3), 'c' is compared with 'a', 'd' is compared with 'b', and 'e' is compared with 'c' and 'd' (if the KeyUp event for 'd' is reported prior to the KeyDown event for 'e'). In case (4), 'd' is compared with both 'b' and 'c'. When a match is found, the second keystroke is marked and the information describing the double letter is stored with it.

Choosing a Key Repeat Delay

Analysis of key press lengths is carried out by the `spotDouble` function whenever an alphanumeric key is raised. Information on average key press length and upward variance in press lengths is recorded. The average key press length is calculated over all the alphanumeric keys that have been pressed since modelling started for this user. The model also explicitly counts the number of keystrokes of each length between 10

and 40 ticks, giving information about the variance in key press lengths for the current user.

The basic key repeat delay recommended is chosen by examining the counts recording press length values and choosing a value which is greater than at least 98% of the user's keystrokes. The maximum of this value and the value given by the formula ((2 * pressLengthAverage) + 3) is chosen as the recommended repeat delay.

To insulate against effects of deliberate long key presses (e.g. a user typing aaaarrrrgggghhhh!!!!!), any keystroke longer than 40 ticks is considered exceptional and not included in the calculations. The number of exceptionally long keystrokes is counted. If more keystrokes are ignored than used, the model recommends that the key repeat facility should be disabled. This may not be the most reasonable reaction - the long keystrokes may be deliberately taking advantage of the repeat facility, using the arrow keys for example. However, in practice, disabling the key repeat facility was never recommended by the model at any time for any typing sample in either the internal or external evaluation (see Chapter 8).

To enable the model to respond quickly to changes in the input stream characteristics (due to a new user arriving, or an existing user changing), a short term key press length average is also kept. This extends over the last 20 keystrokes. If the long and short term averages differ by more than 3 ticks, then the long term average is discarded and replaced by the more recent average. The key press length counts and count of ignored key presses are also updated.

In the initial design for this aspect of the model, it was intended that special keys such as *Delete* and the arrow keys would be excluded from the calculations, and from the count of exceptionally long key presses, owing to the frequency with which auto-repeats are deliberately invoked for these keys. These exceptions were, however, omitted from the implemented model. Ultimately, this omission had little effect on the evaluation because few participants deliberately used the auto repeat on the *Delete* and arrow keys for the tasks given to them. Implementation of these exceptions is expected to improve the robustness of the model in less constrained typing tasks.

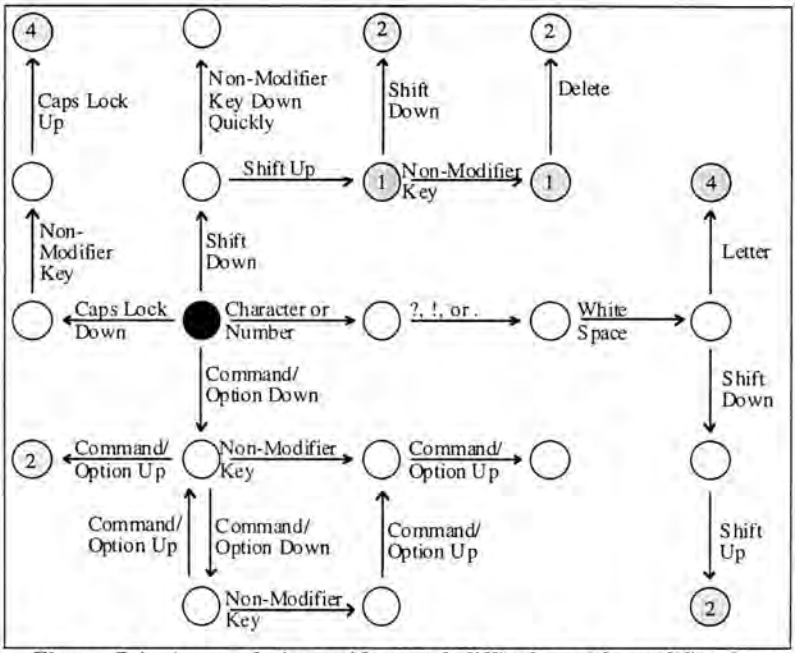


Figure 7.4: Accumulating evidence of difficulty with modifier keys

Assessing the Need For Sticky Keys

As discussed in Chapter 4, people who have difficulty in holding down one key while pressing another often employ strategies to circumvent the problem, or make errors such as dropping errors. They may avoid using *Shift* altogether.

Recognition of difficulties in using modifier keys has been implemented using a state transition machine, illustrated in Figure 7.4. The machine recognises characteristic keystroke patterns. It has eight special states, indicated in light grey, which contribute evidence of difficulty in using modifier keys. Each state contributes 1-4 points to the evidence total, as indicated in the diagram. The starting state is indicated in black. States are linked by transition arcs, each labelled with an event type which allows a transition between the two states.

A number of states may be active at any one time. Initially, only the starting state is active. The basic operation of the machine is as follows:

1. The starting state is placed into the list of active states.
2. The first/next event is read.
3. If the event is a modifier key being pressed down, then update the count of uses of modifier keys.
4. Decay the total evidence value, if necessary. This is done after every 10 modifier key presses. At each decay, if the total is greater than zero, it is reduced by one point.
5. The starting state is placed into the list of new states.
6. For each of the currently active states, if there is a transition from that state to a new state whose label matches the current event, then add the new state to the list of new states.
7. The list of new states becomes the list of active states.
8. For each active state, add any evidence points to the total.
9. If the evidence value has crossed a threshold, update the model's recommendation on the use of *Sticky Keys*. The threshold for recommending *Sticky Keys* is 30 points or over. If the total is 10-29 points, then *Sticky Keys* is suggested. Below 10 points, it is not considered likely to be useful.
10. Go to Step 2, repeat until modelling is terminated.

There is one additional feature of the algorithm not mentioned above. Special behaviour occurs when a modifier key other than *Shift* or *Caps Lock* is pressed. On a Macintosh, these are the *Command* and *Option* keys. While these keys are pressed down, key patterns have a different meaning, and so cannot be interpreted in the same way. For example, the keystroke sequence ". there" would normally be interpreted as evidence that the user was avoiding the use of capital letters. If the *Command* key was pressed down while the '.' was pressed, however, the meaning is quite different, and no evidence should be counted. For this reason, when the *Command* and *Option* keys are pressed, the list of active states is suspended until the key is released. A single new state, as shown in Figure 7.4, is produced. The original states are reactivated only when neither *Command* nor *Option* are down.

The keystroke patterns used as evidence have been derived from the empirical data described in Chapter 4. Certain sources of evidence are dependent on the context in

which the user is typing. For example, Microsoft Word can be set to automatically capitalise the first word in a sentence. With this feature activated, the key sequence ". there " no longer necessarily indicates difficulty in using modifier keys.

There are other potentially useful sources of evidence not exploited by this model. For example, no timing information is taken into account. While time evidence can be unreliable, due to the many alternative explanations possible for any delay, it would be worth investigating if any useful information could be inferred from keystroke timings around the use of modifier keys, compared with those for other keys. Some participants who had difficulties of this kind were observed to press the modified key for an unusually long time, for example.

Interpreting Overlapping Keystrokes

The `spotDrop` routine maintains two counts: one of the number of additional key errors, and one of the number of deliberate overlapping key presses. Only additional key errors involving two or more characters being generated are recognised.

Whenever a non-modifier key is pressed down, the current status of the keyboard, as maintained by the generalInfo routine, is examined. Every other alphanumeric key currently pressed down represents a potential error or deliberate overlap, and all candidates are examined. They are classified as representing an error, a deliberate overlap, or of unknown cause.

In assessing additional key errors, knowledge of the keyboard layout is crucial. The keyboard positions of the two letters are examined, and if they are not adjacent on the keyboard, the pair is judged to be a deliberate overlap, and the appropriate count is incremented.

If the characters are adjacent on the keyboard, then the frequency of the digram they represent is examined. This value specifies, for the first character of the digram, the percentage of times it is followed by the second character in English. Digrams are not available for all character pairs, so where the frequency is not known, the event cause is unknown. If the frequency is known, and is 5% or greater, then the digram is quite likely to occur deliberately, and the event is classified conservatively as of unknown cause. If the digram frequency is 1-5%, then the current user's tendency to deliberately overlap keys is taken into account. If the count of deliberately overlapping keystrokes

is less than 3% of the total number of keystrokes examined so far, then the event is interpreted as an additional key error, otherwise it is of unknown cause. Any digram with a frequency of less than 1% is considered likely to be an error, and the additional key error count is incremented.

The threshold values used in this algorithm were established experimentally, using the data described in Chapter 4. Since the typing samples used in this experiment represented many copies of the same short text passage, generalisability of this approach and the threshold values used cannot be guaranteed. The evaluation described in Chapter 8 addresses this issue to some extent, but further work on more realistic text samples would be necessary to establish the ideal thresholds.

Assessing the Need for Bounce Keys

Every double letter created by two separate key presses is a potential bounce error, and is examined by the `spotBounce` function. This function classifies double letters as deliberate, bounce errors, or of unknown cause. When a bounce error is identified, an evidence value dependent on the uncertainty associated with the classification is calculated. This contributes to an accumulating total of evidence that the user has a tendency to make bounce errors. The evidence decays over time, decreasing by 0.2 every 50 key presses.

When the evidence value becomes greater than 5.0, the *Bounce Keys* utility is recommended, and an appropriate debounce setting is calculated, indicating the number of ticks for which to suppress repeated characters. An ideal setting will eliminate bounce errors while allowing deliberate double key presses to be made without requiring the user to alter their typing patterns. In general, bounce errors are made more quickly than deliberate double letters. An ideal setting may not exist, however, since the timing of the slower bounce errors can be longer than the fastest double letters. The model recommends a conservative setting, choosing a value that will suppress no more than two of the double letters classified as deliberate. (The relevant timing information is stored similarly to that for long key presses.)

Classification of double letters is based on the digram frequency for the character in question, and the length of time between the raising of the key on the first press, and the pressing down of the key on the second press: the *double gap*. Double presses of

Delete and the arrow keys are not considered, since these are likely to be deliberate. Double letters for which digram frequencies are not known are ignored. When a digram frequency is available, this information is combined with the double gap, and the average double gap for keys previously classified as deliberate, to perform the classification.

If the double gap is less than 2/3 of the average double gap for deliberate doubles, and the frequency for that digram is less than 3%, the double is classed as a bounce error. When a bounce error is identified, the contribution it makes to the total evidence is calculated in two parts: the gap contribution and the frequency contribution. The gap contribution is:

gapContribution = 10 * ((doubleGapAverage - doubleGap) / doubleGapAverage)

Because the formula is only applied when the double gap is less than the average double gap, this value is always positive. It represents the proportion of the average gap by which the current gap is lower, scaled to a value between 0 and 10. The smaller the current gap, the greater the gap contribution.

The frequency contribution is calculated as:

frequencyContribution = (3.0 - frequency) / 3.0

This is a similar formula to that for the gap contribution. It represents the difference between the frequency of this double letter in English, and 3% - the maximum value - as a fraction of the maximum value. This produces a value between 0 and 1. The lower the frequency, the greater the frequency contribution. In practice, digrams with a frequency value of less than 1% are not explicitly stored, in order to reduce storage requirements of the model. A frequency of 0% is used for these digrams, producing a gap contribution of 1.

These values are combined as:

(frequencyContribution * gapContribution * gapContribution) / 10

to produce a number between 0 and 10. There is not a linear relation between the distance of a gap below the average gap, and the likelihood of it being a bounce error. The gap contribution is therefore squared in order to reduce the effect of low and mid-range values, emphasising that of cases in which the observed gap between the double letters is much less than the average gap between double letters for that person. This

value is then multiplied by the frequency contribution. This has no effect for letters where the frequency was less than 1%, since the frequency contribution is 1, but reduces the value for those with frequencies between 1% and 3%. The larger the frequency, the greater the reduction in the evidence provided by this double letter. The resulting value is then scaled by dividing by 10. This produces the evidence value for the current double letter, which is added into the total evidence accumulated.

If the double gap is within 1 tick of the average, or greater than the average, and the frequency of that digram is 3% or more, the double is classified as deliberate. If neither of these criteria apply, the double is of unknown cause. Its timing information is recorded as if it were a double. This means that the double gap average may be lower than the true value, and that the *Bounce Keys* setting chosen may also be lower than the ideal value, contributing to the conservative nature of the algorithm.

Keyboard Layout and Digram Information

The model uses keyboard layout information and a database storing the frequency with which a given character is followed by another given character in modern English. The representations used are motivated by the need to provide fast access while minimising storage requirements.

Keyboard layout information is organised around particular keys on the keyboard, rather than ASCII characters. Keyboard adjacencies are represented by an array of 96 * 3 integers, where each set of three integers represents the neighbour information for one key and is treated as an array of 96 bits. Up to 96 keys can be represented, but generally only 62 or 63 are used. A function exists to map a given character to an index in the range 0-95. Characters sharing the same key on the keyboard (e.g. 'a' and 'A') will map to the same position in the array, avoiding redundant representation of information. Within the entry for a single character, the array of 96 bits is indexed in the same way as the main array. Where a character with index i is adjacent to another character with index j on the keyboard, bit j after position i in the array is set to one. If the characters are not adjacent, the bit is zero. Extracting neighbour information is then a case of finding the indices of the two characters, and examining the appropriate bit in the integer array. Different keyboards can be handled by replacing this array. It would be possible to implement a dynamic solution, capable of handling a number of different keyboards which may even be connected at the same

time, but this would require an increase in storage space, and code specific to the Macintosh architecture. The current implementation contains no Macintosh-specific code in the model routines.

The digram frequencies were calculated from the British National Corpus, which contains over 100 million words, representing many different varieties of English. (More information is available at: http://info.ox.ac.uk/bnc.) The digram information could be replaced or supplemented with similar statistics about languages other than English, or any command or programming language. There are 180 digrams with a frequency greater than or equal to 1% to be represented. The raw digram counts were calculated from the British National Corpus, and kindly provided by David McKelvie and Chris Brew from the Cognitive Science department at the University of Edinburgh.

Representation of digram frequencies is more complex. The language specific information is encapsulated in two arrays which can be altered or replaced when other languages are required. Here, the distinction between 'a' and 'A' is important, so specific characters, rather than keys on the keyboard, must be represented. In order to reduce memory requirements, only those digrams with a frequency of 1% or greater are explicitly represented. For each of these, the exact frequency is recorded. The representation could be further reduced by raising the threshold and omitting storage of the actual values. This representation was chosen to allow greater flexibility in manipulating the parameters and algorithms of the model.

Of the 180 digrams to be represented, there are 80 different initial characters. A `FrequencyCount` array lists, for each of the 80 initial characters, the number of digrams which start with that character, and gives an index into a second `FrequencyList` array. The `FrequencyList` array stores pairs consisting of the ASCII value of the second character of the digram, and the frequency value itself. To find the frequency of a given digram 'XY', a function is used to extract an *index* associated with the first character[7] - X. If no index is returned, the digram has a frequency of less than 1%, and a value of 0 is returned. The *count* and *index2* values stored at `FrequencyCount[`*index*`]` are used to access the *count* entries in the FrequencyList array, starting at *index2*. The character Y is matched against the ASCII

[7] If a language other than English were used, then this function would also be replaced, to reflect the new array sizes and mapping from ASCII characters to array indices.

value in the array. If it matches, the frequency value stored with it is returned. If none of the entries match, then the digram is uncommon and the frequency returned is zero.

7.4.3 Output: Presenting Results

The control panel displays the current state of the model for each of the four aspects of typing examined, indicating the current evidence value where relevant. The values displayed are dynamically updated in line with the current state of the model.

In addition, a log file can be produced, detailing the input events examined, and the activity of the model over time. These log files are similar to those generated by *InputLogger*. An example log file is shown in Appendix B.3.

7.4.4 Static UNIX Version

The internal evaluation of the model was based on pre-recorded log files of typing data, therefore a different, UNIX-based, version of the model was used to access these log files. The version differs in that the input module simply reads lines from the log file, and the output module prints results to the screen when the end of the input file is reached. The central modelling facility is identical to that used in the Macintosh version.

7.5 Internal Evaluation of the Model

In the initial implementation of the model, guesses were made as to appropriate values for parameters such as evidence threshold levels, constants in the repeat delay and bounce evidence formulae, and weightings associated with different patterns of modifier key usage. Tuning of these parameters was based on an internal evaluation using the typing data gathered in the study described in Chapter 4.

Forty-four typing log files from twenty participants with disabilities and six with no disabilities were available. The format of these files has been specified in Appendix B.3. All were copies of the same text passage. Some included error correction and some did not.

These files were used to simulate direct computer input, with the events being read from a text file instead of the input event queue. When the whole log file had been read, the state of the user model was examined.

For long key press errors, additional key errors and bounce errors, the model's ability to identify users having difficulty was assessed by comparing the number of errors occurring with the number recognised by the model. For *Repeat Keys* and *Bounce Keys*, the accuracy of the model's configuration recommendations was assessed by examining the number of errors occurring in each typing test, and comparing this with an estimate of the number of errors that would have occurred had the recommended configuration been used. For modifier key usage, a more sophisticated approach was required, to take into account the coping strategies adopted by users who found it difficult to press two keys at once. Assessment of the model in this area was based not only on error numbers, but also on the user's reported and observed ease of using both hands, and their preference (if known) for using *Sticky Keys*.

The model is text-independent, but because this data consisted of many copies of the same text passage, the internal evaluation of the modelling techniques did not provide an insight into the model's potential performance over more general English text. This issue was addressed more fully in the external evaluation, which used four different text passages, and is described in Chapter 8.

Iterations of parameter adjustment and evaluation were carried out until the results were acceptable. While the results could have been further improved by extending the model to handle some of the idiosyncrasies observed in the data, this may have compromised the generality of the model, and was not done. Alterations to constant values already existing in the model were the only changes made during internal evaluation. The following section describes the results that were achieved by this process in each of the four sections of the model.

Table 7.1: Internal evaluation of key repeat delay chosen

Participant	Reported difficulty	Setting chosen		Errors remaining		Avcrage key press length
		(T1)	(T2)	(T1)	(T2)	
1	easy	15	18	1	0	7
2	easy	12	.	0	.	5
3	easy	21	21	4	5	9
4	easy	11	11	1	2	4
5	easy	19	30	11	13	6
6	easy	24	22	12	10	10
7	some difficulty	24	25	11	11	11
8	easy	22	.	15	.	12
9	easy	19	21	9	12	9
10	hard	38	.	5	.	17
11	easy	11	11	3	2	4
12	easy	35	32	1	0	16
13	very hard	41	.	0	.	20
14	hard	22	.	1	.	10
15	hard	21	23	2	0	10
16	easy	12	12	0	0	5
17	easy	26	.	17	.	10
18	easy	20	.	0	.	9
19	extremely difficult	34	36	1	0	16
20	easy	24	23	1	1	10
C1	easy	19	.	0	.	8
C2	easy	11	12	0	0	5
C3	easy	11	11	0	0	4
C4	easy	12	12	0	0	5
C5	easy	10	11	0	0	4
C6	easy	13	13	0	0	5

7.6 Final Results of Internal Evaluation

7.6.1 Repeat Keys

Long key press errors were the most common type of difficulty found in the original study, and choosing an appropriate setting for the key repeat delay is for many the single most important mechanism for improving keyboard usability.

Because the model had precise information about the length of all alphanumeric key presses, counting long key press errors for any given key repeat delay was trivial. Long key press error numbers have been presented in Chapter 4. The results

presented here predict the errors that would have occurred, had each participant been using the precise setting recommended by the model.

Table 7.1 shows the results of the model for the twenty participants with disabilities (1-20) and the comparison group (C1-C6). The table shows each participant's reported level of difficulty in making quick key presses, the repeat delay setting (measured in ticks) recommended by the model at the end of each of their two typing tasks (T1 and T2), the number of long key press errors that would have occurred in each task had the recommended repeat delay been in force for the whole of the time spent typing, and the participant's average key press length.

The recommended delays ranged from 10 to 41 ticks, and the maximum number of long key press errors remaining in a single task was 17, for Participant 17, which represents a 2.8% error rate. In the original data, the maximum error rate for a default repeat delay of 16 ticks was 66.6%, for Participant 13.

Six of the participants reported some difficulty in making short key presses, and indeed the three participants for whom the longest repeat delays were suggested were among this group. A total of 17 participants, including one from the comparison group (a novice computer user), were advised to use key repeat delays longer than the default.

One interesting result was that for Participant 5, who rated short key presses as 'easy'. In both her typing tasks the average key press length was 5 ticks, but the variation among key press lengths was large. She made many long key presses, particularly in the second task (fatigue may have contributed to the difference). Because the recogniser took this variation into account, long repeat delays were advised in order to cope with the longer key presses she sometimes made. A recogniser based purely on average key press lengths would be unable to accommodate participants with similar wide variations in their key press lengths.

The projected total number of long key press errors for all participants using their recommended setting was 151, as opposed to the 2610 projected errors under a default key repeat delay. This represents a greatly improved individual configuration for the participants studied.

Table 7.2: Modifier key difficulty recognition

Participant	Reported difficulty	Dropping errors		*Sticky Keys* evidence		Recommended setting		Ideal setting
		T1	T2	T1	T2	T1	T2	
1	impossible	0	3	47		on	on	on
2	easy	0	.	7		off	.	on
3	easy	0	1	0	0	off	off	off
4	moderate	8	6	25	19	maybe	maybe	maybe
5	hard	0	0	1	1	off	off	on
6	impossible	2	3	70	60	on	on	on
7	easy	4	0	22	4	maybe	off	maybe
8	easy	0	.	2	.	off	.	off
9	some difficulty	0	2	3	6	off	off	maybe
10	very hard	0	.	71	.	on	.	on
11	impossible	0	0	84	48	on	on	on
12	some difficulty	4	2	54	53	on	on	maybe
13	easy	3	.	14	.	maybe	.	maybe
14	easy	0	.	0	.	off	.	off
15	hard	11	2	36	61	on	on	on
16	easy	0	0	9	0	off	off	off
17	easy	0	.	0	.	off	.	off
18	easy	0	.	43	.	on	.	off
19	moderate	0	0	11	0	maybe	off	maybe
20	moderate	2	2	58	14	on	maybe	on
C1	easy	0	.	3	.	off	.	off
C2	easy	0	0	0	0	off	off	off
C3	easy	0	0	0	3	off	off	off
C4	easy	0	0	0	0	off	off	off
C5	easy	0	1	3	7	off	off	off
C6	easy	0	0	0	0	off	off	off

7.6.2 Sticky Keys

The results of modelling difficulties in the use of modifier keys are summarised in Table 7.2. The table shows, for each participant, the difficulty they reported in performing multiple key presses, the number of dropping errors they made in each typing task (T1 and T2), the final total of accumulated evidence of a need for *Sticky Keys* in each task (T1 and T2), the *Sticky Keys* setting recommended by the recogniser ('on', 'off' or 'maybe') for each task, and the 'ideal' setting for each participant. This last value was arrived at by considering the participants' reported level of difficulty, their preferred configuration, and the dropping errors they made. *Sticky Keys* was ideally 'on' for any participant who usually used it, or reported finding modifier key use at least 'hard'. The ideal recommendation was 'off' for participants who expressed no difficulty in using modifier keys, never used *Sticky*

Keys, and made no more than one dropping error per passage typed. The ideal recommendation was 'maybe' for all other participants.

The configuration recommended for one or both of the typing tasks agreed with the 'ideal' configuration for 21 of the 26 participants. Of the five cases where the recogniser made a less than ideal choice, three (Participants 2, 5 and 9) were participants who either used *Sticky Keys*, or might have found it useful, but for whom it was not recommended. All of these participants were able to use modifier keys competently, but typed predominantly with one hand. The recogniser could not detect how awkward or tiring an action may have been for the user, and could only judge their actual performance. In the fourth case, for Participant 12, *Sticky Keys* was recommended, while the ideal recommendation was 'maybe'. A year after the original data was gathered, the same person also participated in the external evaluation (as Participant ED4). At that time he had started to use *Sticky Keys*, and considered it essential. The model's recommendation is therefore reasonable.

In the remaining case, that of Participant 18, the model mistakenly recommended the use of *Sticky Keys*. This recommendation was based on the observation that Participant 18 used the *Caps Lock* key for all single capital letters. This was actually due to a lack of understanding of the keyboard, rather than difficulty with modifier key presses. Cases such as this may be common, and the possibility that the user does not understand the use of *Caps Lock* or *Shift* should be considered by any system interpreting these recommendations.

Overall, the performance of the model was good. The use of *Sticky Keys* was recommended for all those who rated modifier key presses as 'very hard' or 'impossible'.

7.6.3 Bounce Keys

Seven participants made up the total of 44 bounce errors. The results of bounce error detection are shown in Table 7.3. The use of *Bounce Keys* was recommended in five of the eleven tasks in which at least one bounce error occurred. Six bounce errors were on the *Caps Lock* key, including all three of the errors in Participant 15's second typing task. Bounce errors on *Caps Lock* could not be detected by the model, or eliminated by the use of *Bounce Keys*.

Table 7.3: Bounce error recognition

Participant	*Bounce Keys* setting		Bounce errors		Bounce evidence	
	T1	T2	T1	T2	T1	T2
1	off	off	0	0	0.0	0.0
2	off	.	0	.	0.0	.
3	off	off	0	0	1.8	0.9
4	off	off	0	0	0.0	0.0
5	off	off	0	0	0.0	0.0
6	4	2	8	12	20.8	6.0
7	off	off	0	0	0.0	0.9
8	off	.	0	.	0.0	.
9	off	off	0	1	0.6	2.9
10	off	.	0	.	0.0	.
11	off	off	0	0	1.9	0.0
12	off	off	2	1	0.0	1.0
13	9	.	3	.	5.7	.
14	off	.	0	.	0.0	.
15	6	off	3	3	14.9	0.0
16	off	off	0	0	0.2	0.0
17	off	.	0	.	4.5	.
18	off	.	0	.	0.0	.
19	off	off	3	0	1.4	0.0
20	6	off	5	3	15.9	0.2
C1	off	.	0	.	0.3	.
C2	off	off	0	0	0.0	1.3
C3	off	off	0	0	0.0	0.0
C4	off	off	0	0	0.2	0.6
C5	off	off	0	0	0.4	0.0
C6	off	off	0	0	0.0	0.0

Bounce Keys was not recommended for any participant who did not make bounce errors. However, in the case of Participant 17, a high level of spurious evidence (4.5) had been gathered. A longer typing test would be necessary to reveal whether this evidence would decay into insignificance, or whether similar spurious evidence totals would develop for other participants.

The *Bounce Keys* settings chosen by the model varied between 2 and 9 ticks. Imposing these recommended delays on reactivation of keys would have eliminated 19 of the 38 bounce errors not on the *Caps Lock* key. They would also eliminate 4 deliberate key presses. It is difficult to separate deliberate key presses from bounce errors, and so the results here seemed a good compromise - losing one deliberate key press for every five errors eliminated.

Table 7.4: Additional key error recognition

Participant	Reported difficulty	Problem indication		Additional errors		Number detected		Spurious errors		Deliberate overlaps
		T1	T2	T1	T2	T1	T2	T1	T2	
1	moderate	yes	yes	29	29	23	24	0	2	no
2	easy	no	.	0	.	0	.	0	.	no
3	some difficulty	no	no	5	4	2	2	0	0	yes
4	some difficulty	no	no	3	0	1	0	0	1	no
5	some difficulty	no	no	2	0	2	0	0	0	no
6	easy	no	no	0	1	0	1	0	0	no
7	some difficulty	no	no	3	0	0	0	0	0	no
8	easy	no	.	3	.	1	.	0	.	no
9	easy	no	no	2	0	2	0	0	1	a little
10	easy	no	.	3	.	2	.	0	.	no
11	easy	no	no	3	1	3	0	0	0	no
12	easy	no	no	2	0	0	0	0	0	no
13	easy	no	.	0	.	0	.	0	.	no
14	some difficulty	no	.	0	.	0	.	0	.	no
15	some difficulty	no	no	5	7	3	3	1	0	no
16	easy	no	no	1	0	0	0	0	0	no
17	easy	no	.	4	.	4	.	0	.	no
18	easy	no	.	0	.	0	.	0	.	no
19	moderate	yes	no	18	11	8	3	1	2	yes
20	easy	yes	yes	14	9	10	8	0	0	no
C1	easy	no	.	0	.	0	.	0	.	no
C2	easy	no	no	1	0	0	0	3	2	yes
C3	easy	no	no	0	2	0	0	0	0	a little
C4	easy	no	no	1	0	1	0	1	1	a little
C5	easy	no	no	0	0	0	0	1	0	no
C6	easy	no	no	0	0	0	0	0	0	no

7.6.4 Additional Key Errors

Additional key errors were relatively common in the data available, and usually involved two key presses which overlapped in time. Table 7.4 shows, for each participant, their reported ease of isolating keys to press (Reported Difficulty), whether a problem with additional key errors was found by the model (Problem Indication), and the actual numbers of additional key errors made in the tests (Additional Errors).

A problem was identified for three participants. These were the three participants who made the most additional key errors, including the two who reported the most difficulty. The majority of the remaining participants did make some additional key errors, but their error rates were low.

The table also shows the number of genuine errors that were detected (Number Detected) - 63% of those present in the original data. Errors were missed most frequently for those participants who often deliberately overlapped keystrokes. These are indicated in the final column of the table (Deliberate Overlaps). For these participants, particularly Participant 19, additional key errors were difficult to distinguish from normal typing. The error detection mechanism was designed to be conservative, in order to avoid detection of errors where none exist. Nevertheless, 16 spurious errors (Spurious Errors) were found, also shown in the table.

To allow for spurious errors, a rate of one or two errors in every 100 characters should be tolerated when making decisions based on the model's results. Error rates above this threshold should cause a problem to be flagged. For some users who make additional key errors but do not deliberately overlap key presses, the *Overlap Keys* utility outlined in Chapter 6 may provide a useful level of support.

7.7 Summary

The typing data from the study described in Chapter 4 has provided a sound basis for the development of a model of keyboard skills and configuration requirements. Keyboard, as opposed to mouse difficulties, and the configuration facilities which address them, have been chosen as the domain for this model because keyboard errors are easier to recognise than mouse errors, and also because there is a greater range of configuration facilities available to address these difficulties. Mouse usage may require a different approach. This issue will be discussed in Chapter 9.

Given the proposed application of the model in the field of adaptive systems outlined in Chapter 3, a number of requirements can be identified. The ideal model should be individual, implicit, dynamic, and short-term. It should produce recommendations as quickly as possible without making demands on the user's time and energy. While the field of user modelling has produced many powerful techniques, outlined in Chapter 3, none of the existing general purpose methods were appropriate for this problem. This

was often due to their reliance on task knowledge, or to the qualities of the input data being inappropriate. A domain specific approach has therefore been taken.

The resulting model addresses four areas of keyboard difficulty: use of modifier keys, long key press errors, bounce errors and additional key errors. In addition to recognising users prone to each of these difficulties, it makes recommendations for settings of the appropriate existing configuration facilities: *Sticky Keys*, *Repeat Keys* and *Bounce Keys*. The model operates by trapping and examining keystrokes as they are typed by the user, and the only assumption made about the user's input is that it is English text. Users' typing can potentially be assessed while they are performing an unknown text composition task - no specific demands are made of users. This is one feature of the model that might be of particular interest to the field of user modelling as a whole, where explicit user questioning or testing are often used, and can introduce difficulties of their own (see Chapter 3, Section 3.2.2 for a discussion of the disadvantages of building models by questioning users). A related feature of the model is the lack of restriction on the type of input that can be interpreted. While the model presented here operates on English text input, the same principles could be applied to any textual input with language-like regularities from which meaningful digram frequencies could be extracted.

Internal evaluation of the accuracy of the model has been carried out using the data from the typing tasks described in Chapter 4. These forty-four recorded typing logs were used to simulate dynamic input, and the model's conclusions at the end of each complete log file have been examined. The results have been used to tune the model parameters, without compromising the generality of the model. If the configuration recommendations made by the model were to be applied to the original log files, the number of long key press errors would have been reduced from 2610 to 151. Use of *Sticky Keys* where recommended could have eliminated 54 of the 56 dropping errors, and would have helped nine of the eleven participants who reported difficulty in using modifier keys. All three participants prone to additional key errors were recognised, as were four of the seven who made bounce errors. Importantly, the model only once recommended an inappropriate facility for a user, and this was prompted by the user's misunderstanding of the use of modifier keys.

While these results are encouraging, they are hardly surprising. The model's algorithms were based on the same data as used in the internal evaluation - the generalisability of this data is unknown. The evaluation is also limited in that all the

log files contained the same text passage. While the model's design is text independent, it remains to be shown that it will perform adequately on arbitrary English text. In addition, this analysis has examined the model's response only at the end of each input passage. The speed of response of the model, and its stability, are also important factors to be examined. These issues will be addressed in the following chapter, which describes the external evaluation of the model.

Chapter 8

Evaluation of the Model

The previous chapter has described the development of a user model of four aspects of typing difficulty. The model was based on, and tuned using, the typing data described in Chapter 4. In order to assess the model, it was necessary to test it on different users (both with and without typing difficulties) and different text input. The responsiveness and stability of the model in a dynamic setting were also to be investigated.

This chapter describes the model's external evaluation, which took the form of an empirical study, using a similar methodology to that developed in Chapter 4. These results have appeared in condensed form as Trewin and Pain (1998). The primary goal of the external evaluation was to assess the performance of the model. This was measured by comparing the model's recommendations with the keyboard difficulties exhibited by the participants, and with the participants' opinions of the utilities suggested by the model. Secondary goals of the evaluation were to investigate the time taken by the model to draw conclusions about a user's typing, and to examine the stability of the model's recommendations over a period of typing. These two tasks were complicated by the fact that a user's typing may itself be changing over a period of time.

This chapter does not include a discussion of the relative performance of the model with subjects with and without disabilities. Because the model examines four separate aspects of typing, and not all participants with disabilities had difficulty in all four of these areas, such a distinction would be false.[1] Instead, the focus is on the model's performance for participants with and without each of the specific difficulties examined

[1] It is theoretically possible that the model is, for example, accurate at modelling people with disabilities but inaccurate for people without disabilities. Comparison of the model's performance on these two groups would then be necessary. As will be seen, the high quality of the results presented in this chapter does not invite such comparisons.

by the model. All of the subjects without disabilities had no physical difficulty in operating the keyboard. Of the subjects with disabilities, the majority experienced just one or two of the problems under examination.

As in Chapter 4, typing data was recorded for participants copying text. Four different passages were used. A detailed description of the experimental procedure is given in Section 8.3. In brief, participants copied an initial passage with the model activated. They then tried up to three further passages, each using one of the configuration facilities covered by the model, and gave their opinions of these facilities. The model remained activated throughout the session, in order to provide stability information over as long a typing session as possible. The errors made, and participants' initial and final opinions, were then compared to the model's recommendations.

As a side-effect, the evaluation process also produced data on the effectiveness of *Repeat Keys*, *Sticky Keys*, and *Overlap Keys*, which have been described in Chapters 5 and 6.

8.1 Participants

Thirty participants took part in the evaluation. Twenty had a motor disability affecting their use of the keyboard (Participants ED1 to ED20), while ten had no motor disability (Participants EN21 to EN30). The participants have been described previously in Chapter 5, pages 106-112. Disabilities included muscle control difficulties and spasms (5 people), cerebral palsy (4 people), incomplete tetraplegia (3 people), nervous system damage (3 people), effects due to stroke (3 people), multiple sclerosis (1 person) and rheumatoid arthritis (1 person). The effects of these disabilities on keyboard use included tremor and spasm in the hands and fingers, co-ordination difficulties, loss of dexterity, weakness and pain when pressing keys.

The table summarising the experience, disability and typing style of each participant is reproduced here as Table 8.1 for ease of reference. There was no significant difference between the disabled and non-disabled groups in terms of age ($t=-1.669$, $p=0.106$) or experience level ($t=-0.162$, $p=0.872$).

Table 8.1: Participant summary

Participant	Experience (years)	Disability affecting typing	Typing style	Number in previous study
ED1*	34.0	constant movement and spasms in the hands and fingers	left	15
ED2*	0.4	difficulty in controlling hand and finger movement without visual attention	left	5
ED3*	28.0	pain when making keystrokes, constant movement and occasional spasms	both	19
ED4*	7.0	control of hand movement requires effort, movement slow	right	12
ED5	0.2	loss of dexterity in hands and wrists, movement stiff and painful	both	.
ED6*	2.0	no use of left hand, difficulty in making precise, aimed movements	right	1
ED7	1.5	loss of dexterity in left hand	mainly right	.
ED8	0.0	movement control and force difficulties, some spasm	both	.
ED9	0.1	pronounced hand shake, particularly left hand	both	.
ED10	5.0	difficulty and pain when moving wrists and fingers	both	.
ED11*	28.0	difficulty in straightening fingers, some hand spasms	both	3
ED12	0.2	spasm in hands and arms, particularly when fatigued	both	.
ED13*	29.0	impaired hand dexterity	mainly right	4
ED14	0.04	tremor and spasms in hands	mainly left	.
ED15	2.0	right hand unsteady and shaky	mainly left	.
ED16	1.02	right hand unusable	left	.
ED17*	6.0	co-ordination difficulties, muscle spasms	both	13
ED18	7.0	semi-paralysis on left side and weakness on right, very painful to touch any object	mainly right	.
ED19*	0.8	limited use of left hand, difficulty flexing fingers of right hand	mainly left	2
ED20	12.0	difficulty in controlling movement and spasm in hands	both	.
EN21	2.4	none	both	.
EN22	9.0	none	touch	.
EN23	0.3	none	both	.
EN24	14.0	none	both	.
EN25	2.8	none	touch	.
EN26	18.0	none	both	.
EN27	0.0	none	mainly right	.
EN28	7.0	none	both	.
EN29	22.0	none	touch	.
EN30	0.0	none	both	.

Five of the participants with disabilities typed with one hand only, while six typed predominantly with one hand but could use the other hand for modifier key presses. The remaining nine typed with both hands (not touch typing). Of the ten participants with no relevant disability, one typed mainly with one hand, six typed with both hands, and three were novice touch typists.

Nine of the participants (ED1, ED2, ED3, ED4, ED6, ED11, ED13, ED17 and ED19) had previously provided data used during model development and internal evaluation. In this chapter, a '*' will be used to indicate members of this group when referring to individual participants. Their number in the previous study is shown in the final column of Table 8.1.

8.2 Materials

As described in Chapter 5, four matched text passages were used. Each required 625 keystrokes, and included 21 capital letters and 9 punctuation marks requiring the use of the *Shift* key. Each passage contained one sequence of four capital letters for which use of the *Caps Lock* key was appropriate (although not essential). A standard form was used to record information about each participant and their session. This form, and the four text passages, are reproduced in Appendix A.2 for reference.

Macintosh 475 8/160 and Power Macintosh 6100/66 machines, and the SimpleText word processor were used. The Macintosh version of the model described in Chapter 7 was used. This version of the model produces a log file similar to that produced by *InputLogger*, which contains information about the evidence used by the model to draw conclusions and the time at which conclusions were drawn, in addition to a record of the `KeyDown` and `KeyUp` events which occurred. The format of the log file is shown in Appendix B.3.

8.3 Procedure

Experimental sessions were limited to two hours, and extended only if the participant chose to continue. Participants were informed that they were free to stop or rest at any time. Each session was video recorded (the camera being focused on the keyboard)

and logged by the model itself. The same observer (the author) administered each session.

Background information about the participant, including their disability and level of keyboard experience, was recorded by the experimenter on the form shown in Appendix A.2.

For each of *Sticky Keys*, *Repeat Keys*, *Bounce Keys*, and *Overlap Keys*, the participant was asked if they had heard of the facility. Those who had were asked to describe its operation, in order to verify that their understanding was correct. For those who had not heard of a facility, or did not describe it accurately, it was described verbally, and where further explanation was required a demonstration was given. For the *Bounce Keys* utility, no demonstration was available, since it was not supported on the Macintosh platforms used in the experiment. Participants were not given the opportunity to try the facilities. The participants' previous awareness of the utility, and knowledge of whether it was available on the machine they normally used, were recorded. Participants were deemed to be aware of a facility if they had known such a thing existed, regardless of whether they had recognised the name.

Participants were then asked how useful the facility would be to themselves, and how often they would use it. Responses to the former question were given on the scale: 'not useful', 'somewhat useful', 'useful', 'very useful', or 'essential'. For the latter question, responses were on the scale: 'never', 'rarely', 'sometimes', 'often', or 'always'. For *Repeat Keys*, participants were not asked about frequency of use, since some key repeat delay is always used. Instead they were asked to give their preferred setting for the key repeat delay, if known.

The model was activated. Participants were then asked to copy one of the passages as accurately as possible using the default keyboard configuration. They were free to make corrections if they wished, or to ignore their errors. A variety of approaches to error correction were anticipated, providing information about the model's response to more natural editing actions. Negative transfer of learning effects are to be expected for participants used to a different keyboard configuration. For *Repeat Keys* and *Sticky Keys*, these effects have been discussed in Chapter 5. No participant normally used any other software facility.

Participants were then asked to copy up to three further passages, each with one of *Sticky Keys*, *Repeat Keys*, or *Overlap Keys* enabled. Constraints on the time

available, and the limited stamina of some participants, meant that many did not complete four passages. The time taken to type the first passage was used to give an initial prediction of how many further passages could be typed in the time available. In choosing the order of activation of the utilities, those which had been recommended by the model or thought useful by the participant themselves were given priority over those which had not been indicated as potentially useful. A further order constraint was imposed by the fact that activation of *Sticky Keys* affected the input events recorded for modifier keys, and hence the model's view of the need for *Sticky Keys*. In order to provide information on how the model behaved over several passages of typing, activation of *Sticky Keys* was reserved for the last passage typed as often as possible. In practice, some participants typed further passages after *Sticky Keys* had been tried, because they felt able to continue. The majority of participants tried facilities in the order: *Repeat Keys*, *Overlap Keys*, *Sticky Keys*.

The order in which the text passages were presented was varied between participants, in order to counteract any passage-specific effects.

The experimenter activated and deactivated the facilities each time, and informed the participant of what had been changed, and the effect on the keyboard. When *Sticky Keys* was activated, participants were told that they should generate capital letters by pressing and releasing *Shift* exactly once, and then pressing the desired character, or by using *Caps Lock* if preferred.

After having used each facility, participants were again asked to rate the usefulness of the facility, and frequency with which they would use it, using the same scales as previously. The responses serve as a subjective measure of the quality of the model's recommendations, supplementing objective measures based on input errors.

When all facilities had been tried, the time was exhausted, or the participant chose to stop, the model was deactivated. They were then asked whether, and at what point, they had experienced pain or fatigue during the passages. The video recording was then stopped.

All participants typed passages in a single session, with breaks between passages, with the exception of Participant ED8, who typed two passages on two separate sessions, having become too tired to continue after the initial passage on the first session.

8.4 Analysis

As described in Chapter 4 and Chapter 5, the observations made and video evidence were used to manually annotate the recorded log files, indicating types of errors made and time spent correcting errors of each type. Any observed difficulties in using the utilities were also annotated. The annotations used, and an example fragment from an annotated log file, are given in Appendix B.4.

The annotated log files were then automatically filtered and the Systat statistical package (SYSTAT, 1992) used to perform the analyses. Nonparametric statistics were used, as the variables under examination do not have, or cannot be assumed to have, normal distributions.

8.5 Results

A number of different assessment criteria have been used, not all of which are suitable for all four areas tackled by the model. These criteria are listed below, along with the areas for which they are appropriate.

- **Identification of input errors.** Where the model is designed to identify instances of specific performance error classes, it can be assessed according to how many of the errors present were detected, and how many non-errors were incorrectly considered to be errors. This is the main assessment criteria used for bounce errors and additional key errors.

- **Effect of suggested configuration on input errors.** Where a specific configuration setting intended to reduce errors is suggested by the model, it can be assessed according to its effectiveness in eliminating these errors. For *Repeat Keys*, the suggested key repeat delay is assessed in this way - participants type a passage using the suggested delay and the errors occurring are examined. For *Bounce Keys*, participants could not try using the facility, so assessment is based on a projection of the number of errors observed that would have been eliminated by the proposed setting. This approach is not appropriate for *Sticky Keys*, since many users who find modifier keys awkward to use do not make input errors when using them.

- **Participants' opinions of the suggested facilities.** Where participants tried a configuration facility, their opinion of the facility provides additional information on the accuracy of the model's suggestions. This is the primary assessment criteria used for *Sticky Keys*. It is also used for *Repeat Keys*. While this subjective measurement is not ideal (as discussed in Section 8.6.2), it is ultimately the user who decides whether to adopt a given configuration option, and so a good correlation between the model and users' opinions would be a positive result.

- **The stability and responsiveness of the model.** For *Repeat Keys*, *Sticky Keys*, and *Bounce Keys*, the model makes configuration recommendations which may change over time. Where a participant's requirements are stable, the model's recommendations should also be stable, and this stable state should be reached as quickly as possible.

Use of the model had no noticeable effect on response time of the word processor.

8.5.1 Repeat Keys

The model recommends a specific key repeat delay, measured to the nearest tick. However, on the Macintosh architecture the only settings available are 12, 16, 24 or 40 ticks, or suppression of repeats altogether. For the purposes of evaluation, the model's precise recommendation was therefore transformed into the nearest available setting at or greater than that recommended by the model. For example, if the model recommended a delay of 13 ticks, the setting chosen would be 16 ticks. Chapter 5 has shown that *Repeat Keys* is effective in reducing long key press errors. The purpose of this evaluation is to show that the chosen setting was near optimal for the user, where an optimal setting is one which eliminates long key press errors while still allowing keys to repeat as quickly as possible.

As described in Chapter 7, there was a bug in the implementation of the model used during the evaluation, which meant that the arrow keys, *Delete* and *Return* were included in repeat delay calculations. The log files recorded were analysed by a debugged version of the model to extract the results which should have been produced. For some of those participants who made deliberate use of the repeat facility on these keys, the results observed during the evaluation had been strongly influenced by

periods of arrow key usage, resulting in instability of the model's conclusions. In the majority of these cases, arrow keys were pressed for longer than the participant's typical alphanumeric key press length. Some participants, however, pressed arrow keys for less time than alphanumeric keys, resulting in a shorter delay than would have been produced by the bug free version of the model.

Fortunately, in only four cases were the differences between the observed and corrected repeat delays sufficient to warrant a different repeat delay setting. Of these four cases, two did not try any altered repeat delay. The remaining two had actually tried the corrected delay, either in the first or second passage. Twenty-three participants tried using the recommended corrected delay, and nine of these participants also tried the setting below the corrected setting. Since the delay settings tried by the participants matched the corrected delays well, the debugged version of the model is evaluated here, using the corrected delay values.

Effect of Chosen Setting

When examining the settings recommended for *Repeat Keys*, each participant would ideally have been asked to type with the key repeat delay setting closest to that suggested by the model, and also with higher and lower settings. However, restrictions on the time available for the evaluation sessions meant that only a single text passage could be used. While it would have been possible to split the text passage into three sections of approximately thirty-five words, and use a different repeat delay setting in each section, these sections may have been too short to allow users to adjust to the new delay. Instead, participants typed the whole passage with the recommended delay. Where the recommended delay was 16 ticks, participants had already tried their ideal setting while typing the initial passage. In these cases, participants tried a lower delay of 12 ticks, allowing some investigation of whether the model's recommendations were overly conservative. Twenty-three participants tried using the recommended delay, and eleven of these participants also tried the setting below their recommended setting. Ideally, there should be very few long key press errors using the recommended setting. More errors would be expected when using the lower setting.

Table 8.2 shows the twenty-three participants who completed a passage using the setting at or above the corrected value recommended by the model. For each

Table 8.2: Repeat delay evaluation

Participant	Model's choice of setting after first passage	Nearest setting at or above model's choice	Nearest setting below model's choice	Error rate with setting at or above model's choice (%)	Error rate with setting below model's choice (%)	Error rate with default setting of 16 ticks (%)
ED3*	28	40	24	0.0	.	18.9
ED8	30	40	24	0.0	.	21.0
ED9	20	24	16	0.3	2.1	2.1
ED10	16	16	12	0.2	.	0.2
ED11*	22	24	16	0.0	1.1	1.1
ED12	16	16	12	1.4	.	1.4
ED13*	12	12	.	0.0	.	0.0
ED14	16	16	12	1.2	.	1.2
ED15	16	16	12	2.3	.	2.3
ED17*	40	40	24	0.3	.	74.4
ED18	12	12	.	0.8	.	0.0
ED19*	11	12	.	0.0	.	0.0
ED20	19	24	16	0.0	0.6	0.6
EN21	14	16	12	0.2	0.2	0.2
EN22	17	24	16	0.0	0.2	0.2
EN23	14	16	12	0.0	1.0	0.0
EN24	10	12	.	0.0	.	0.0
EN25	15	16	12	0.0	0.4	0.0
EN26	11	12	.	0.0	.	0.0
EN27	14	16	12	0.3	.	0.3
EN28	11	12	.	0.0	.	0.0
EN29	13	16	12	0.0	0.0	0.0
EN30	13	16	12	0.0	0.2	0.0

participant, the model's recommendation and the settings above and below the recommendation are shown. The long key press error rate observed when using these settings, and the error rate observed with the default setting, are also listed.

After typing a single passage, the recommendations made by the model for these participants varied between 10 and 40 ticks. Overall, the model's recommendations reduced the average error rate from 5.4% to 0.3% over the twenty-three participants who tried an altered delay. When using the nearest available *Repeat Keys* setting at or above the recommended value, fourteen of the twenty-three participants made no errors.

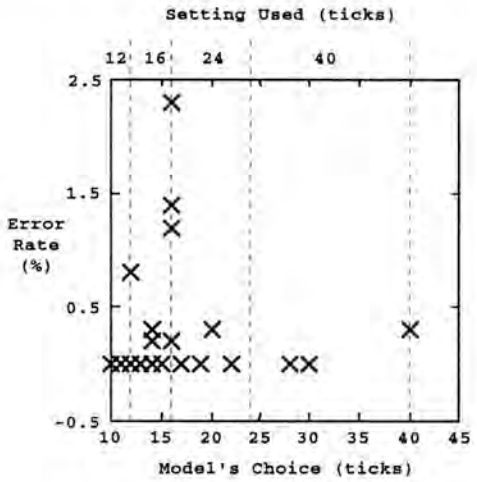


Figure 8.1: Error rates with altered delay

Of the nine who did make errors using the recommended setting, four had error rates greater than 0.5%, the maximum being 2.3%. Figure 8.1 illustrates the relationship between the model's recommended repeat delay and the error rate observed for these twenty-three participants. The vertical axis represents their error rate with the new delay, while the lower scale on the horizontal axis represents the model's recommendation. Vertical dashed lines divide the plot according to the delay setting actually used, as indicated at the top of the plot. In six of the nine cases where the new error rate was non-zero, the participants were using the exact setting suggested by the model.

Nine participants also tried the setting below that recommended by the model. It was anticipated that if the model's recommendations were near optimal, errors should appear when using the setting below the recommended setting. In eight of these cases, errors were observed using the lower setting. With the exception of participant EN21, who made only one long key press error in each passage, the error rates for these eight participants were indeed significantly higher when using the lower setting ($p = 0.017$, Wilcoxon matched pairs signed ranks test).

User Opinions

Participants were asked their preferred key repeat delay setting both before and after using the default and altered settings. Twenty-five participants expressed an initial preference and fifteen expressed a final preference. Six participants changed their preferred setting, two increasing it, and four decreasing it after having tried an altered key repeat delay. The Spearman Rho indicated a correlation of 0.559 ($p<0.05$) between participants' initial and final preferences. The recommendation of the model after the first text passage showed significant correlation with the participants' final opinions (Spearman Rho = 0.723, N=15, $p<0.01$), but no significant correlation with participants' initial opinions (Spearman Rho = 0.225, N=24).

Stability and Responsiveness

Table 8.3 shows information about the stability of the model's recommendations, within the context of the five available settings on the Macintosh. It gives the dominant setting chosen for each participant, the percentage of keystrokes for which that setting was recommended, the number of changes in the model's recommendation after an initial stabilisation period of twenty keystrokes, and a description of the changes in the model's choice over time.

In relation to the available Macintosh settings of 12, 16, 24 and 40 ticks, the model's choice of setting was stable for over 99% of the typing of eighteen of the thirty participants. All passages typed were included in this calculation. For all of these eighteen participants, the model stabilised within twenty keystrokes.

For Participants ED1* and ED16, the model recommendation stabilised more slowly, resulting in a single recommendation accounting for over 97% of their typing. Of the remaining ten participants, Participants ED12, ED17* and ED18 showed an increase over time in the repeat delay recommendation by the model, which corresponds with a significant increase in their key press lengths over time, previously noted in Chapter 5. For Participants ED13*, ED14, EN23 and EN27, the model's recommendation decreased over time. For participants ED2*, ED5, and ED15, the model recommendation did not appear to stabilise, switching between two adjacent settings, or in the case of Participant ED2*, between three settings.

Table 8.3: Stability of the model's recommendations

Participant	Dominant setting	% of keystrokes covered by dominant setting	Number of changes after initial stabilisation period	Description of behaviour over time
ED7	12	100	0	stable after 1st keystroke
ED19*	12	100	0	stable after 1st keystroke
ED20	24	100	0	stable after 1st keystroke
EN28	12	100	0	stable after 1st keystroke
EN25	16	100	0	stable after 2nd keystroke
EN21	16	100	0	stable after 2nd keystroke
EN26	12	100	0	stable after 4th keystroke
ED11*	24	99.9	0	stable after 5th keystroke
EN24	12	99.9	0	stable after 5th keystroke
EN22	24	99.9	0	stable after 6th keystroke
ED3*	40	99.9	0	stable after 2nd keystroke
EN29	16	99.9	0	stable after 3rd keystroke
ED8	40	99.8	0	stable after 3rd keystroke
ED10	12	99.6	0	stable after 18th keystroke
EN30	16	99.5	0	stable after 17th keystroke
ED6*	24	99.5	2	settled on 24 after 1st keystroke, with one 12 stroke period of 40
ED4*	40	99.3	2	settled on 40 after 1st keystroke, with one 12 stroke period of 24
ED9	24	99.1	4	settled on 24 after 9th keystroke, with two < 10 stroke periods of 40
ED1*	24	97.6	8	varied between 24 and 40, then settled on 40 after 133 keystrokes
ED16	24	97.1	10	varied between 16 and 24, then settled on 24 after 77th keystroke
EN23	16	96.5	4	settled on 16 after 7th keystroke, with a 1 and an 84 stroke period of 24
EN27	16	94.0	1	24 for first 77 keystrokes, then 16
ED13*	12	81.0	1	settled on 16 after 1st keystroke, changed to 12 after 274th keystroke
ED14	16	75.1	21	varied between 16 and 24, then settled on 16 after 414th keystroke
ED12	24	68.0	30	varied between 16 and 24
ED17*	40	66.0	5	settled on 40 after 9th keystroke, with three < 10 stroke periods of 'Off' before settling on 'Off' after 619th keystroke
ED5	24	56.9	10	varied between 24 and 40
ED15	16	56.9	3	varied between 16 and 24, then settled on 16 after 551st keystroke
ED18	16	53.1	3	settled on 12 after 1st keystroke, with one 1 stroke period of 16, changed to 16 after 1166th keystroke
ED2*	12	51.4	24	varied between 12, 16 and 24

Using twenty keystrokes as the typical time for the model to adjust to a participant's typing, Table 8.3 shows the number of changes in the model's recommendation for each participant after the first twenty keystrokes. The mean number of changes was 4.27 and the median 0.5. On average, the model's recommendation changed every 451 keystrokes. 39.8% of these recommendation changes were of very short duration, with the previous choice being reinstated or a new choice chosen within ten keystrokes.

8.5.2 Sticky Keys

The model accumulates evidence that the current user may have difficulty in using modifier keys, using the technique described in Chapter 7, Section 7.4.2. Initially, the *Sticky Keys* recommendation is 'no'. When the evidence level reaches a threshold of 10 points, the value is set to 'maybe'. When the evidence level reaches or exceeds 30 points, the recommendation given is 'yes'. The evidence value decays in relation to the number of times modifier keys are used.

User Opinions

Table 8.4 summarises the evidence values and recommendations produced by the model at the end of transcription of the first text passage by each participant. The table also shows the number of dropping errors they made in the passage, and the participants' opinion of *Sticky Keys* after having tried it.

The algorithm imposes a lower limit of 0, and no upper limit on evidence values. The values observed after the first passage ranged from 0 up to 356.

Twenty-five participants expressed an initial opinion other than 'don't know' about *Sticky Keys*. Twenty-five participants tried using *Sticky Keys*, and twenty-two of these expressed a final opinion other than 'don't know'. Ten participants revised their opinion after having tried the utility, as reported in Chapter 5. There was a significant correlation between the participants' initial and final opinions (Spearman Rho = 0.678, N=22, $p < 0.01$). The evidence value calculated by the model showed significant correlation with both the initial (Spearman Rho = 0.405, N = 25, $p < 0.05$) and final (Spearman Rho = 0.480, N = 22, $p < 0.05$) opinions of the participants.

Table 8.4: *Sticky Keys* results

Participant	Evidence	Recommendation	Dropping errors	Participant's final opinion
ED1*	50	yes	2	useful
ED2*	57	yes	1	very useful
ED3*	0	no	0	.
ED4*	63	yes	2	essential
ED5	1	no	0	.
ED6*	356	yes	31	very useful
ED7	15	maybe	0	very useful
ED8	0	no	0	.
ED9	9	no	2	don't know
ED10	2	no	0	useful
ED11*	1	no	0	.
ED12	49	yes	0	very useful
ED13*	17	maybe	3	useful
ED14	0	no	0	very useful
ED15	9	no	0	essential
ED16	73	yes	0	very useful
ED17*	6	no	3	.
ED18	4	no	0	not useful
ED19*	7	no	1	useful
ED20	7	no	0	useful
EN21	7	no	0	not useful
EN22	12	maybe	0	somewhat useful
EN23	2	no	1	useful
EN24	0	no	0	somewhat useful
EN25	6	no	0	useful
EN26	4	no	0	useful
EN27	10	maybe	0	don't know
EN28	1	no	0	very useful
EN29	0	no	0	not useful
EN30	0	no	0	.

Table 8.5: *Sticky Keys* recommendations

Participant's final opinion	Model's recommendation		
	Yes	Maybe	No
Essential	ED4*		ED15
Very useful	ED2*, ED6*, ED12, ED16	ED7	ED5, ED14, EN28
Useful	ED1*	ED13*	ED10, ED19*, ED20, EN23, EN25, EN26
Somewhat useful		EN22	ED3*, EN24
Not useful			ED11*, ED18, EN21, EN29

Table 8.5 illustrates the match between the model's recommendations and the opinions of the twenty-five participants who had tried *Sticky Keys* (either prior to, or during the session) and expressed some opinion about it. One-handed typists are shown with double underlining, predominantly one-handed typists have single underlining, two-handed typists have dotted underlining, and touch typists have no underlining.

The table shows that all five of the one-handed typists considered *Sticky Keys* at least 'useful', and all were given a 'yes' recommendation by the model. Of the six mainly one-handed typists, five thought the utility at least 'useful', and the model recommended 'maybe' for only two of this group - three potential users were missed. Turning to the two-handed and touch typists, the results initially appear worse: nine of the eleven who rated the utility at least 'somewhat useful' were given 'no' recommendations. However, closer examination of the data gives a different picture. Of these nine false negatives, two considered the utility useful only because it seemed to them to provide a simpler way to generate modified characters. The remaining seven considered the utility useful because there were occasions when they did type with one hand, for example when using the telephone. None had any physical difficulty in using modifier keys in the default way, which is what the model is intended to detect.

Stability and Responsiveness

The recommendation made by the model remained stable over all passages typed without *Sticky Keys* for twenty-four of the thirty participants. Table 8.6 shows the evidence values accumulated and the model recommendations for the passages typed consecutively without *Sticky Keys*. Activation of *Sticky Keys* alters the input stream in such a way as to eliminate most sources of evidence. The model's values therefore decayed slightly when *Sticky Keys* was activated.

The values suggest a general accumulation of evidence over time for some participants, particularly those with high values after the first passage. This caused the model's recommendation to alter for six participants. In the most extreme of these cases, that of Participant EN22, one dropping error in the second passage and four in the third passage contributed to the accumulation. In the case of Participant ED20 and Participant EN23, the change in recommendation brought the model closer to the participant's opinion. Participant EN22 also considered *Sticky Keys* 'somewhat

Table 8.6: Model changes over passages typed

Par-ticipant	Passage 1 evidence	Passage 2 evidence	Passage 3 evidence	Passage 1 conclusion	Passage 2 conclusion	Passage 3 conclusion
ED1*	50	109	.	Yes	Yes	.
ED2*	57	126	188	Yes	Yes	Yes
ED3*	0	4	7	No	No	No
ED4*	63	117	.	Yes	Yes	.
ED5	1	3	.	No	No	.
ED6*	356	460	.	Yes	Yes	.
ED7	15	21	.	Maybe	Maybe	.
ED8	0	3	.	No	No	.
ED9	9	14	.	No	Maybe	.
ED10	2	.	.	No	.	.
ED11*	1	0	0	No	No	No
ED12	49	46	.	Yes	Yes	.
ED13*	17	18	.	Maybe	Maybe	.
ED14	0	.	.	No	.	.
ED15	9	.	.	No	.	.
ED16	73	.	.	Yes	.	.
ED17*	6	8	.	No	No	.
ED18	4	12	20	No	Maybe	Maybe
ED19*	7	6	.	No	No	.
ED20	7	12	15	No	Maybe	Maybe
EN21	7	9	14	No	No	Maybe
EN22	12	22	35	Maybe	Maybe	Yes
EN23	2	4	10	No	No	Maybe
EN24	0	0	4	No	No	No
EN25	6	5	2	No	No	No
EN26	4	2	7	No	No	No
EN27	10	.	.	Maybe	.	.
EN28	1	2	4	No	No	No
EN29	0	3	.	No	No	.
EN30	0	0	.	No	No	.

useful'. Participants ED18 and EN21 did not consider it useful, while Participant ED9 did not know whether it would be useful.

Where the model's recommendation was 'yes' or 'maybe' at the end of the first passage, the conclusion was reached after the participant had typed between 4 and 21 modified characters, the average being 13.2.

8.5.3 Bounce Keys

Bounce error numbers are used to identify participants who may have wished to try *Bounce Keys*. Because the *Bounce Keys* utility was not available for participants to

Table 8.7: Bounce errors and the model's recommendations

Participant(s)	Bounce error rate (errors per 100 correct characters)	Total bounce errors made	Final evidence value	Final choice for *Bounce Keys* (ticks)	Total double letters classed as errors	Number of bounce errors identified
ED9	3.03	68	25.71	3	5	26 (38.2%)
ED1*	0.52	11	13.87	4	2	4 (36.4%)
ED3*	0.36	9	16.05	5	4	5 (55.6%)
ED17*	0.21	3	8.50	7	0	3 (100%)
ED20	0.16	5	0.00	.	0	0 (0%)
ED11*	0.14	3	0.00	.	5	1 (33.3%)
ED12	0.09	2	0.00	.	5	1 (50%)
ED14	0.08	1	4.51	.	1	1 (100%)
EN27	0.07	1	0.00	.	0	1 (100%)
ED2*	0.07	2	0.00	.	0	0 (0%)
ED5	0.06	1	0.00	.	4	1 (100%)
ED4*	0.05	1	0.00	.	2	0 (0%)
ED13*	0.04	1	0.09	.	7	0 (0%)
EN26	0.04	1	0.00	.	5	0 (0%)
EN22	0.04	1	0.00	.	0	0 (0%)
EN25	0.03	1	0.00	.	0	0 (0%)
EN28	0.00	0	0.00	.	10	n/a
EN21	0.00	0	0.37	.	5	n/a
EN23	0.00	0	1.9	.	1	n/a
ED16, EN24	0.00	0	0	.	1	n/a
ED6*, ED7, ED8, ED10, ED15, ED18, ED19*, EN29, EN30	0.00	0	0.00	.	0	n/a

try, and only two of the participants had previously tried the utility, it is not possible to assess the model's performance in terms of its correlation with the participants' opinions. Instead, the specific *Bounce Keys* setting chosen by the model is assessed in terms of the number of bounce errors it would have eliminated, and the number of deliberate double letters that would have been affected by that setting.

Error Identification

Table 8.7 shows the participants ranked according to the rate at which they made bounce errors over all passages typed, with the highest rates shown first. The table gives the bounce error rate and total number of bounce errors made by each participant,

the final evidence value calculated, and the recommended debounce time after all passages had been typed. A '.' indicates that the model did not suggest the use of *Bounce Keys*. The table also shows the number of deliberate double letters mistakenly considered bounce errors by the model, and the number and percentage of genuine bounce errors that were recognised as such by the model.

Bounce error rates observed over all the passages typed were very low: twenty-four of the participants made less than one bounce error per 1000 characters. The highest error rate observed was 3.03 errors per 100 correct characters, for Participant ED9 (representing 68 actual errors). The *Bounce Keys* threshold could reasonably be raised so that *Bounce Keys* was recommended for Participant ED9 only. Further research into the use of *Bounce Keys* is required to establish the typical error rate at which *Bounce Keys* is useful.

The evidence threshold at which *Bounce Keys* is suggested is 5.00. The model suggested *Bounce Keys* for the four participants with the highest bounce error rates (0.21 errors per 100 correct characters, or more): ED9, ED1*, ED3* and ED17*. During typing of the passages, *Bounce Keys* was also suggested for Participant ED14, but the evidence value had decayed below the threshold by the end of the experiment. *Bounce Keys* was never suggested for any other participant.

The model typically identified 1/3 of a participant's bounce errors correctly, and, on average, wrongly labelled 2 double letters as bounce errors. In no case did the incorrect labelling of double letters as bounce errors cause the model to recommend *Bounce Keys* for a subject who did not make bounce errors. The threshold and decay values, and the bounce evidence formula itself, which chooses evidence values according to the model's confidence that the current double letter is a bounce error, were effective in preventing such misdiagnoses. Even in the case of Participant EN28, where 10 double letters were misclassified, the evidence associated with these 'errors' was so low that by the end of the simulation it had decayed to zero, and it never accumulated over the threshold value.

In all of the first three passages typed, there was a strong positive correlation between the evidence value calculated by the model, and the number of bounce errors made by the participant. In the first passage, Spearman's Rho measured the correlation as 0.766 ($p<0.01$). In the second, the correlation was 0.667 ($p<0.01$) and in the third it was 0.563 ($p<0.01$). The correlation decreased over the passages, and broke down in the final passage, due to the very small numbers of bounce errors made, and the

increasing effect of the initial evidence value at the start of the passage, since the evidence accumulated over all passages typed. Enough bounce errors were correctly identified to ensure that the model recommended *Bounce Keys* for the four participants most prone to bounce errors (having error rates > 0.2%). The facility was also suggested for one other participant, for a period of 800 characters. This participant (ED14) made just one bounce error. The model did not recommend *Bounce Keys* for any participant who did not make bounce errors.

Effect of Chosen Setting

For those participants for whom *Bounce Keys* was suggested, Table 8.8 shows the delay suggested, the number of bounce errors they made, and the number of those errors that would have been eliminated had the final proposed *Bounce Keys* setting been imposed. The table also shows the number of deliberate double letters typed by each participant, and the number of these that would also have been suppressed by the proposed *Bounce Keys* setting.

Table 8.8: Effect of proposed *Bounce Keys* settings

Participant	*Bounce Keys* setting chosen	Total bounce errors	Number suppressed by proposed *Bounce Keys* setting	Total double letters	Number affected by proposed *Bounce Keys* setting
ED9	3	68	22	178	1
ED1*	4/5	11	6/7	96	9
ED3*	5	9	4	179	4
ED17*	7	3	2	98	5
ED14	3/5	1	1	21	0/3

For all of these participants, many more deliberate double letters than bounce errors were observed. The *Bounce Keys* settings chosen were low, and (with the exception of Participant ED14, who made only one bounce error) eliminated 32.4% to 66.7% of the bounce errors observed. The settings would not have affected more than nine deliberate double letters for any participant (including use of the arrow keys, and the *Delete* key), but in three cases (using the larger value recommended for Participant ED14) would have affected more deliberate key presses than bounce errors. Reducing the delay suggested by the model would reduce the number of double letters affected by the utility, but would also reduce the number of errors eliminated.

Stability and Responsiveness

The responsiveness of the model is summarised in Table 8.9 for those participants for whom *Bounce Keys* was recommended at some point. The table shows the number of bounce errors made by each participant before *Bounce Keys* was suggested, and the number of characters that had been typed up to that point. The total number of bounce errors and characters typed are also shown. The final column of the table summarises the model's recommendations between the time that *Bounce Keys* was first suggested, and the end of the experiment.

Table 8.9: Model sensitivity and stability

Par-ticipant	Bounce errors before *Bounce Keys* suggested	Characters typed before *Bounce Keys* suggested	Total bounce errors made	Total characters typed	Changes from first recommendation to end of typing (character: recommendation)
ED9	7	36	68	2246	36: delay 3
ED1*	2	521	11	2107	521: delay 4 700: no delay 756: delay 4 1417: delay 5 2071: delay 4
ED3*	4	995	9	2525	995: delay 5
ED17*	2	1324	3	1406	1324: delay 7
ED14	1	430	1	1290	430: delay 5 641: delay 3 1241: no delay

It took between one and seven bounce errors before *Bounce Keys* was recommended for these participants. For participants with low error rates, many characters were typed before this recommendation was made. For three of the participants, the recommendation remained unchanged after having been made. For Participant ED14, who made only one bounce error (in the first passage) and typed two passages, the evidence value decayed sufficiently that *Bounce Keys* was no longer recommended by the end of the second passage. For Participant ED1*, who typed three passages, *Bounce Keys* was initially suggested after two bounce errors had been made. The evidence then decayed until it fell below the threshold after 5 bounce errors had been made, near the end of the first passage. On the sixth bounce error, typed 56 characters later near the start of the second passage, the evidence once again exceeded the

threshold, and *Bounce Keys* continued to be recommended for the remainder of the second and third passages typed.

The specific setting suggested remained stable for three of the participants. For Participant ED14, the initial setting was based on a single bounce error, and was revised to a more conservative value when further double letters had been observed. Both settings chosen would have eliminated the single bounce error made.

8.5.4 Additional Key Errors

This analysis examines the accuracy with which the model recognises additional key errors. The number of errors recognised is expected to be less than the total number of errors, since the model cannot identify those errors in which the intended key did not register, or those in which both characters registered, but the key presses did not overlap in time.

The model also counts overlapped key presses on non-adjacent keys. This information is intended to give an impression of the likelihood that the *Overlap Keys* utility described in Chapter 6, or some similar utility, might be relevant to the current user. Because the efficacy of utilities like *Overlap Keys* remains to be shown, this aspect of the model cannot be assessed in terms of correlations with the opinions of participants who tried *Overlap Keys*.

Error Identification

Measuring additional key error rate as the number of errors per 100 correctly typed characters, excluding errors in which the intended key was not activated, and including errors in which the two key presses did not overlap in time, the highest error rate observed was 2.94% (100 observed errors), for Participant ED6*. A total of eight participants had error rates greater than 0.5% (ED1*, ED3*, ED5, ED6*, ED7, ED20, EN24 and EN25).

Table 8.10 shows the participants ranked according to the rate at which they made such errors, measured as the number of errors per correctly typed character over all the passages typed. The table also gives totals over all passages typed for the number of

Table 8.10: Additional key error recognition

Participant	Error rate (errors per 100 correct characters)	Number of errors	Number and percentage recognised	Number of spurious errors
ED6*	2.94	100	64 (64%)	5
ED20	1.89	59	29 (49%)	2
ED5	1.27	22	16 (73%)	2
ED1*	0.90	19	5 (26%)	2
ED7	0.85	18	11 (61%)	0
E25	0.63	20	11 (55%)	1
ED3*	0.59	15	8 (53%)	1
E24	0.55	16	5 (31%)	8
E26	0.45	12	8 (67%)	2
ED11*	0.42	9	3 (33%)	1
ED13*	0.42	11	7 (64%)	1
E28	0.31	8	6 (75%)	0
ED10	0.21	4	2 (50%)	0
ED2*	0.20	6	4 (67%)	0
ED19*	0.19	4	1 (25%)	0
E23	0.10	3	2 (67%)	5
E29	0.10	2	1 (50%)	0
ED12	0.09	2	2 (100%)	0
ED9	0.09	2	1 (50%)	0
ED15	0.08	1	0 (0%)	0
EN30	0.08	1	0 (0%)	0
ED18	0.07	2	0 (0%)	0
E21	0.07	2	0 (0%)	2
E27	0.07	1	0 (0%)	0
ED4*	0.05	1	1 (100%)	0
ED8, ED14, ED16, ED17*, E22	0.00	0		0

these errors made by each participant, the number and percentage detected by the model, and the number of non-errors incorrectly identified as errors by the model.

Errors in which the two keystrokes did not overlap in time, and errors which generated a digram common in English cannot be recognised by the model. The proportion detected is therefore dependent on the text being typed, and the participant's error patterns. For the sixteen participants who made more than two errors, the model identified 25% to 75% of the errors. In total, 55% of the errors were detected. Over all the participants, there was a strong correlation between the total counted by the model, and the actual errors occurring (Spearman Rho = 0.947, $p < 0.01$). The model also sometimes mistook a deliberate letter pair as an additional key error. The

maximum number of spurious errors identified for any participant was eight. Over all the participants, twenty-seven had more genuine errors than spurious errors counted by the model. There was a positive correlation between the number of additional key errors detected by the model and the number of spurious errors found (Spearman Rho = 0.609, $p < 0.01$).

8.6 Discussion

For the copy typing tasks examined, the model was able to identify the participants with the greatest difficulty in all four of the areas studied, and to make good predictions of the configuration that would suit them. The use of the model for less constrained word processing tasks will be discussed in Chapter 9, Section 9.4.1. The evaluation also showed encouraging results on the stability of the model in general.

However, many of the participants with the greatest difficulty had also provided data for model development: two of the three with the highest long key press error rates; the top additional key error maker; and three of the four with the highest bounce error rates. While the model was accurate in identifying the error tendencies of previously unseen users, it requires further testing on new users with high error rates. Furthermore, three of the five one-handed typists had contributed to the model design. Although the model was equally successful in recommending *Sticky Keys* for these and the two new one-handed typists, further evaluation is necessary in order to test the generality of the techniques used. Further discussion of *Sticky Keys* is given in Section 8.6.2.

Three novice touch typists were included in the evaluation, while none participated in the original study. The accuracy of the model's recommendations for this group, is therefore encouraging. The model should also be evaluated by expert touch typists, to examine whether any characteristics of fast touch typing are mistaken for input difficulties.

Some participants found that errors they made on their usual system did not occur under the experimental conditions, while others found that new errors appeared. In comparing the model's recommendations with the difficulties exhibited by the participants, it is not important whether these errors were due to the different keyboard or computer being used, the different seating arrangements, or simply the pressure

exerted by the experimental setting. What is important is the model's ability to recognise difficulties, regardless of their underlying cause. However these differences between systems are important when comparing the subjects' opinions of the utilities with the model's recommendations. One subject who normally used a system with a key repeat delay longer than sixteen ticks initially considered *Repeat Keys* irrelevant to his needs. After trying to use the default Macintosh key repeat delay, he considered it essential. This could play a part in the improved correlation between the model's recommendations and the participants' opinions after trying the facilities, in comparison with their opinions before trying the utilities. It may also be that some participants found it difficult to judge the usefulness of a facility without having tried it.

It is also possible that the participants' opinions were influenced by the model's results, hence the good correlation between the model and participants' opinions. This point will be discussed further in Chapter 9, Section 9.2.1.

8.6.1 Repeat Keys

For participants with long key press error rates greater than 10% under the default setting, the model's recommendation reduced these rates to less than or equal to 0.3%. In making repeat key delay recommendations, the model showed significant correlation with the opinions of the participants after they had tried the default and recommended settings. The high level of correlation suggests that the model's recommendations were acceptable to the participants.

However, since the majority of the participants did not try the settings above and below their recommended setting, their opinions do not provide information on whether the setting chosen was optimal. This would require a fuller evaluation in which participants tried a number of different settings. A small number of participants did try the setting below that recommended by the model. For the majority of these participants, error rates rose significantly with the lower setting, but the difference was not great. These participants were using very short settings. A more marked effect would be expected at the higher end of the scale of available settings, but this data was not available from the evaluation study and further research is required to confirm this theory, and to provide information on the optimality of the model's recommended setting.

Some information on the accuracy of the specific setting chosen by the model can be gleaned from the observation that all of the participants with error rates over 0.5% when using the recommended setting were using the exact value chosen by the model. Some of these errors may be attributable to a period of adjustment to the new setting. Nevertheless, these error rates suggest that the model's specific recommendations do not eliminate all errors: a higher recommendation may be an improvement.

The model was quick to settle on a reasonable recommendation, typically stabilising within twenty keystrokes. In general, the stability of the model was also good. Some participants were on the border between two settings. For this group, the model was unstable. The current results could be improved by disallowing changes in the model recommendation until the new setting has been consistently recommended for the last ten keystrokes. This would eliminate 39.8% of the changes in recommendation observed in the evaluation.

8.6.2 Sticky Keys

In general, the recommendations made by the model on the use of *Sticky Keys* correlated well with those of the participants themselves, both before and after having tried the facility.

The model never recommended *Sticky Keys* for a participant who thought it 'not useful', but failed to recommend it for twelve who did think it at least 'somewhat useful'. However, the majority of this group were two-handed typists who anticipated using the facility only when temporarily typing with one hand. Only three found modifier keys awkward to use, none of whom had provided data for model development. These three were able to accurately use modifier keys in the default way, and could not be identified by the model. There may be input cues not captured by the model which would have allowed identification of these participants. Alternatively, user questioning or human observation may ultimately be a more accurate approach in this aspect of configuration.

In retrospect, participants' opinions were not a good measure for assessment of the model's recommendations - participants should have been asked about their physical ease of use of modifier keys rather than their general opinion of the utility.

The stability of the model was generally good - changes of recommendation were rare. The number of uses of modifier keys made by the participants before the final recommendation was reached was small. The model could therefore identify those participants who make errors in the use of modifier keys fairly quickly, depending on the number of modified characters in the text they were typing.

For one participant, an evidence value many times higher than the threshold was reached. Allowing such large values has the effect of making the model slow to respond to changes in the user's behaviour, or changes in the current user. Future implementations of the model should impose a ceiling on the evidence value, to improve the responsiveness of the model in such situations.

For at least some of the participants who had no difficulty in making modified key presses, the *Sticky Keys* evidence values accumulated over passages, rather than stabilising. One reason for this may be the artificiality of the task. Matching the number of capital letters in the text passages sometimes introduced unintuitive capitalisation. For example, one passage discussed "the Green party", while another reported on "the new President". Participants had been instructed to reproduce the capitalisation of the passage shown. Occasionally they made wrong assumptions about capitalisation which caused evidence to be accumulated. For example, some participants pressed down the *Shift* key after typing "Green ", and then released it, realising that "party" was not capitalised in the passage. Pressing of modifier keys without any key being modified is one of the evidence patterns used by the model. It is therefore possible that some of the evidence accumulated was an artefact of the tasks themselves.

Other evidence gathered, in particular for Participant EN22, was due to indecision about which *Shift* key to use - she was a novice touch typist. Sometimes she pressed one of the *Shift* keys, and then released it in favour of the other key. The model has no knowledge of which of the two *Shift* keys was pressed each time, and so the input presented was mistaken for dropping errors. The general accumulation of evidence over time suggests that a higher decay value on the evidence may be appropriate. Any increase should, however, be small, in order to retain the stability of the model's recommendations for participants for whom *Sticky Keys* was useful.

8.6.3 Bounce Keys

The accuracy and stability of the model over the periods of typing observed during evaluation were good. The threshold was effective in eliminating noise due to inaccuracies in discrimination between bounce errors and deliberately typed double letters.

Due to the low bounce error rates observed during evaluation, the model typically observed a long period of typing before ascertaining a need for *Bounce Keys*, in comparison with the period required for *Repeat Keys* or *Sticky Keys*. The present implementation of the model recommended the use of *Bounce Keys* consistently for the four participants with the highest bounce error rates. However, three of these highest rates were still relatively low, and the delays recommended in some cases would have affected more deliberate key presses than errors. In practice, it is possible that only the participant with the highest error rate would have benefited from using *Bounce Keys*. Empirical research is required in order to identify the range of error rates for which *Bounce Keys* is typically useful. This information could be taken into account in a future implementation of the model, and the threshold value adjusted accordingly.

Given that for three of the subjects, the specific delay chosen meant that more double letters were affected than errors eliminated, it may be desirable to reduce the recommended delay. However, this would also reduce the number of errors eliminated. Without specific research on the effect of imposing different delay values for users of *Bounce Keys*, it is not possible to evaluate the model's specific delay recommendations in terms of likely user satisfaction. In terms of error reduction, balanced against the requirement to minimise the number of deliberate double letters that would have been affected by the proposed setting, the model's recommendations appear reasonable.

8.6.4 Additional Key Errors

The model detected 55% of the 340 additional key errors observed in the evaluation. It wrongly identified 32 deliberate key presses as errors. There was a strong, significant correlation between additional key error rates and the model's count of additional key errors. This shows that despite the uncertainty associated with classification of

overlapping adjacent keystrokes the model is accurate enough to be useful in identifying those users who are prone to such errors. The values for the relevant model parameters that were chosen during internal evaluation generalised well to the different text passages used in the external evaluation, and to the new participants. The model's results could be used to measure a user's tendency to make additional key errors, and together with the deliberate overlap rates, this information could be used to recommend a utility like *Overlap Keys*. Some of these users may benefit from utilities like *Slow Keys*, or from using a keyguard to reduce additional key errors.

A correlation between the error rates of the participants and the rate at which spurious errors were recorded was also observed. This cannot be explained by the algorithm design - previous error rate is not used in decisions about a new overlapping character pair. A possible alternative explanation would be that typists who tend to deliberately overlap keys are also more prone to additional key errors[2]. Their overlapping keystrokes on adjacent characters may be misinterpreted as additional key errors. This theory is supported by a significant, but fairly weak, correlation (Spearman Rho = 0.328, $p<0.05$) between number of deliberate overlaps observed and number of additional key errors made in the first text passage for each of the thirty participants. However, there was no significant correlation between the number of deliberate overlaps observed, and the number of spurious errors in the same passage (Spearman Rho = 0.212). A satisfactory explanation for this phenomenon has yet to be found.

8.7 Summary

In summary, the model of keyboard configuration requirements was evaluated in an empirical study involving twenty participants with motor disabilities and ten with no motor disability. Participants copied up to four text passages, each requiring 625 keystrokes. The first passage copied was used to produce some model recommendations. In the remaining passages, the participants tried out different configuration facilities and gave their opinions on the facilities.

[2] They may also be more prone to transposition errors, if such errors are caused by excessive overlapping of keystrokes, although this was not investigated in this study.

The model responded quickly to input errors and other evidence that the user was having difficulty, but also showed good stability over typing periods of up to approximately 500 words.

The key repeat delays chosen by the model varied between 10 and 40 ticks, and reduced the average long key press error rate from 3.33% to 0.27%. The error rate of the subject with the longest key presses was reduced from 42.25% to 0.27%, accounting for much of the observed average effect. In the time available for evaluation, it was not possible for participants to try the settings above and below those recommended by the model. However, the cases where the setting used was exactly that chosen by the model suggest that the model's suggestions may be one or two ticks below the optimum level, for at least some participants. Participants found the chosen settings acceptable. The recommendations made by the model stabilised within 20 keystrokes for fifteen of the participants. For seven others, the recommendation either increased or decreased over time, probably in response to changes in the typing style of these subjects due to fatigue or practice. For three participants, the recommendation did not stabilise.

The evidence gathered indicating a user's requirement for *Sticky Keys* showed a significant correlation with the opinions of the participants themselves. The model's recommendation was 'yes' for all seven of the one-handed typists in the study. For all four of the participants who considered it 'not useful', the model's recommendation was 'no'. The major limitation of the model is its failure to recognise some potential *Sticky Keys* users. Nine participants may have occasionally benefited from using the utility, but had no difficulty in using modifier keys in the ordinary way during the model evaluation. This group illustrate the point that configuration utilities are useful to a broader range of users than their original target group. However, since the goal of this model is to identify users experiencing physical difficulty in using the keyboard, their exclusion is unsurprising. Of greater concern is the model's failure to recognise the genuine difficulties of three participants who found modifier keys awkward to use. This aspect of the model requires further research in order to establish whether these results could be improved, or whether this is a hard limitation of this approach.

The *Sticky Keys* recommendations were stable over all passages typed before using the facility for twenty-four of the participants, but some accumulation of evidence over time was observed. Assuming that the participants' need for *Sticky* Keys was not

increasing over time, this indicates that a faster decay rate might be beneficial in controlling the level of evidence values.

For bounce errors and additional key errors, the model was evaluated by assessing how well it recognised such errors. The model recommended *Bounce Keys* consistently for the four participants with the highest bounce error rates. For a short time, the utility was also recommended for a fifth participant, but this participant made fewer bounce errors and the evidence value decayed below the required threshold after two text passages had been completed. The utility was recommended after between one and seven bounce errors had been made. It was never recommended for anyone who did not make bounce errors, indicating that the decay and threshold values were effective in counteracting misclassifications. The key acceptance delays recommended by the model would have suppressed 32.4% to 66.7% of the observed bounce errors for the five participants. An average of four deliberate double letters would have been affected by the proposed settings for each participant.

The model recognised 55% of the additional key errors that were observed, and there was a strong, significant correlation between the number of errors observed and the number of errors made by the participants.

There was an overlap of nine people between the participant group and the participant group used in the original data gathering study of Chapter 4. This is likely to have positively influenced the results. While the model was accurate in identifying the error tendencies of previously unseen users, it requires further testing on new users with high error rates.

While the evaluation has exposed a number of minor alterations that could improve the model's performance, the overall results show that the identification of configuration requirements from free typing is a feasible proposition, at least in the context of the aspects of keyboard use studied here. This result suggests that it may indeed be possible to provide users with automated, user-adapted support for keyboard configuration.

Chapter 9

Discussion and Conclusions

This chapter discusses the results presented, covering keyboard and mouse usage and input errors, configuration facilities, and the model of keyboard configuration skills. Justification for the approach taken is given, and the limitations of this approach identified. Opportunities for further research into input device configuration are described. The experimental methodologies used are discussed. The quality of the resulting model, and potential extensions to the model are also covered, identifying areas for further work. Finally, potential applications of the model are presented. Again, these represent potentially fruitful future research directions.

9.1 The Use of Keyboards and Mice

The development of keyboards and cursor control devices, including the mouse, has been largely driven by the need for fast, efficient, general purpose ways of providing input to computers. Research into the usability of these devices has primarily focused on expert users. Because of this emphasis on efficiency over ease of use, some features of these input devices require very accurate and consistent motor control, and can be difficult for people with motor disabilities.

A number of specific difficulties with keyboards and mice are reported in the literature (Brown, 1992; Brownlow et al., 1989; Edwards, 1995; Poulson, Ashby and Richardson, 1996; Vanderheiden, 1992). Many of these difficulties take the form of performance errors - input errors due to physical inaccuracy in the manipulation of input devices. Common difficulties and errors include: use of modifier keys, dragging, pointing and clicking with a mouse, doubling errors, additional key errors,

dropping errors, remote errors, and bounce errors. The empirical study described in Chapter 4 examined the use of keyboards and mice by people with motor disabilities in more detail, and all of the difficulties and errors discussed in the existing literature were observed. A comparison group of six non-disabled subjects showed significantly fewer errors, although similar types of error were observed.

9.1.1 Generality of the Study

Because this study contained only twenty subjects with motor disabilities, it is not representative of the population of keyboard and mouse users with motor disabilities as a whole. The enormous variation within this population makes representative sampling extremely difficult. As a result, the observations of this study do not necessarily apply to keyboard and mouse users with disabilities in general, and there may be some important areas of difficulty not represented here. Nevertheless, the study did include participants with a wide range of disabilities, levels of computer experience, and ages, and the findings of this study are very similar to informal discussions of keyboard and mouse difficulties already available in the literature. It is reasonable to expect that the findings of this study are characteristic of many keyboard and mouse users with motor disabilities.

9.1.2 Methodology

To the author's knowledge, this was the first formal study of its kind. As such, it provides a starting point for further studies, which could take advantage of the methodology developed here, and the lessons learned. Such studies could, for example, include closer examination of people with specific disabilities, or specific difficulties which have been observed.

There are many methodological restrictions imposed by the requirements of participants with motor disabilities which restricted the form of both this exploration into performance errors and the subsequent evaluation study. Session times were required to be short, allowing rests between tasks. Breaking tasks between sessions was, in general, not reasonable, since the effort of travelling to each session was often significant, and participants were not paid for their time. This limited the volume of

data that could be gathered from each participant to that which they could produce in a two or three hour period.

Ideally, data would have been gathered from participants in the context in which they normally used computers, using machines with which they were familiar. This would have reduced negative transfer of learning effects, and other difficulties caused by unfamiliarity with the machines and input devices used. This would have required data recording facilities capable of running on several different platforms. Unfortunately, it would not be technically possible to record the desired information to an adequate level of accuracy on some platforms. For example, Windows 3.11 platforms do not support the necessary multitasking. As a result, a single platform and data gathering program was used.

The location chosen for sessions was also important. For participants who used Macintosh machines, sessions were carried out at their usual venue wherever possible. For the remaining sessions, adjustable tables and wrist rests were made available, and proved very useful. A ground floor venue was essential.

The study also highlighted the importance of designing tasks suitable for participants who have cognitive impairments associated with the physical impairments of interest. Visual impairment should also be taken into account - the 18 point font size used in these studies was adequate, a smaller font would not have been. The pilot study with members of the target participant group was extremely valuable in identifying potential problems with the original task materials, and the final version was vastly improved. Further simplification of the tasks is recommended for future studies, since the existing tasks were in some cases open to misinterpretation. Longer initial practice periods would have been beneficial for some participants, particularly those unfamiliar with the mouse. This would, however, have reduced the volume of data recorded for those subjects, due to time limitations on each session. Ultimately, performance errors made by novice mouse users are also interesting, and so were recorded. This placed an additional burden on the analysis process, however, since errors of interest - performance errors - had to be differentiated from errors due to uncertainty about how to use a keyboard or mouse.

In this field, choosing a methodology often requires much compromise between practical considerations and the aspects of computer use under investigation. Although the final data acquired is rarely ideal, such empirical studies are extremely valuable and

informative. The results from the model's summative evaluation, discussed in Section 9.3.3, bear out the effectiveness of basing the model on real data from the outset.

9.1.3 Results

This study identified six notable keyboard performance errors: long key press errors, additional key errors, missing key errors, dropping errors, bounce errors and remote errors. While the comparison group spent on average 1.7% of their time correcting errors of this kind in the second typing task, those with motor disabilities spent on average 7.3% of their time on these corrections. Performance errors are clearly important. Use of the *Shift* key was a notable area in which subjects sometimes had difficulty that was not reflected in input error rates.

The majority of the subjects with disabilities found the mouse difficult to use, and many generally try to avoid mouse operations as much as possible. Pointing and dragging were the most difficult operations. Pointing error rates of up to 47% were observed among those with motor disabilities.

The data gathered in this study could be used by other researchers wishing to investigate these or other aspects of keyboard and mouse use by people with motor disabilities.

9.1.4 Use of Alternative Input Devices

For people who have difficulty in using mice and/or keyboards, there are many alternative input devices available, including switch input, eye gaze tracking, voice recognition and alternative keyboards. Physical ease of use and achievable accuracy are not the only considerations when choosing an appropriate input mechanism. Many additional factors, and conflicts between them, must also be taken into account. Some devices do not allow access to the full functionality of all mainstream applications, some are slow, others expensive. Users may also have strong personal preferences unrelated to their performance with the device itself. For any of these reasons, a computer user may choose to use the keyboard or mouse despite some difficulty in operating them, and despite frequent occurrences of performance errors.

In the study described in Chapter 4, for example, most subjects who had tried alternative pointing devices such as the trackerball preferred the standard mouse, despite experiencing problems in using it. For some people, the convenience of using the default input devices outweighs the frustration caused by difficulties in using them.

9.2 The Use of Software Access Facilities

For those who choose to use the standard keyboard and/or mouse, there are many modifications that can improve their accessibility. These range from external facilities such as arm and wrist rests, keyguards and key latches to software which alters the behaviour of the system. Included in the latter class are macros, accelerated writing systems and device configuration options. Configuration options relax the constraints on physical accuracy required by the devices, while macros and accelerated writing systems reduce the amount of input required. The two approaches are complementary.

The majority of the existing software configuration facilities are designed to reduce or eliminate particular classes of performance error. Appropriate configuration is crucial for many people.

Configuration facilities include *Sticky Keys*, *Repeat Keys*, *Bounce Keys*, *Slow Keys*, and *Mouse Keys*. These originated as third-party software patches but are now available in similar forms on the majority of modern operating systems. While the utility of these configuration facilities is not in doubt, there have been virtually no published studies describing their use.

9.2.1 Methodology

Chapter 5 presented data describing the use of *Repeat Keys* and *Sticky Keys* which was gathered during evaluation of the model, as a side benefit of the investigation. The experimental methodology was not ideally suited to investigation of the use of these facilities. It did not allow for proper measurement of practice or fatigue effects, which are likely to have had an effect on factors such as the subjects' forgetting to use the *Sticky Keys* method of generating modified characters. In practice these effects would be difficult to measure. If subjects were asked to type using the default configuration, then an altered configuration, then the default configuration again, this

would allow some measurement of the effect of fatigue and practice on speed. However, when error numbers are considered, there would be errors due to transfer effects when changing between configurations. In Chapter 5 it has been shown that some users adapt their key press lengths according to the key repeat delay currently in force. Ideally, when changing the delay in an experimental setting, some time to adjust to the new delay should be allowed. Unfortunately, many people with motor disabilities will become unacceptably fatigued if asked to perform long passages of typing. It may, therefore, not be possible to eliminate practice effects without introducing strong fatigue effects at the same time.

The experiment did not examine the usability of the facilities as a whole: participants were not required to activate and deactivate them, or to choose their own settings. These experiments show that the facilities have the potential to be extremely useful, but the results for *Sticky Keys* suggest that users may experience difficulties in controlling them. A thorough investigation of these facilities should include assessment of the interfaces they present.

Many participants, when asked to rate the usefulness of the facilities, gave different replies before and after using the facilities. Often the second reply was the more enthusiastic. This result has implications for designers of facilities intended to provide users with support for configuration. It suggests that providing direct experience of a utility strongly increases the likelihood of its adoption, in comparison to providing only descriptions and demonstrations. However, this result should be interpreted with caution. The experiment was designed to examine the usefulness of these facilities, and participants were aware of this aim. In addition, questions were asked directly by the experimenter. Although the questions were initially asked in the same form, participants often required clarification of the question, introducing the possibility of bias in the way in which questions were explained. This, together with knowledge of the experimental goals, may have biased participants in favour of the utilities. A sounder methodology might eliminate direct questioning by the experimenter in favour of allowing participants to try out facilities and choose their own configuration, perhaps administered by a computer program.

9.2.2 Repeat Keys

The examination of *Repeat Keys* described in Chapter 5 suggested that when an appropriate key repeat delay is used, the numbers of long key press errors occurring can be drastically reduced (in the best case, the error rate was reduced from 74.4% to 0.3%) with no negative side effects. Fourteen of the participants with disabilities and three without a disability reported finding the facility useful, very useful or essential. In the absence of any difficulty in activating and setting *Repeat Keys*, the facility can make an effective contribution to keyboard accessibility for computer users with motor disabilities. *Repeat Keys* was also considered useful by users who had no difficulty in adapting to the default repeat delay. While the methodology used did not eliminate potential biases due to practice or fatigue effects, these have been examined separately and were not found to undermine the effectiveness reported for *Repeat Keys*.

This experiment did not directly examine the effect of altering the key repeat delay on actions which often take advantage of the key repeat facility, such as deletion of a string of characters, or navigation using the arrow keys. Further research would be required to establish the effect of these types of actions on the ideal key repeat delay.

9.2.3 Sticky Keys

Chapter 5 also examined *Sticky Keys*, finding that it was effective in eliminating dropping errors. It was considered useful by 79% of those who tried it, including six non-disabled participants. Many of the latter group reported that there were occasions when they did use the keyboard with one hand. Two participants found the utility essential to their keyboard use.

Unlike users of *Repeat Keys*, *Sticky Keys* users are required to alter the way in which they use the keyboard, by using a different mechanism for generating modified characters. In this experiment, even the novice participants were previously exposed to the default method of generating modified characters, using simultaneous key presses, before trying *Sticky Keys*. This resulted in negative transfer of learning effects when *Sticky Keys* was used. Many participants sometimes reverted to the default method, causing *Sticky Keys* to deactivate. It is quite likely that the majority of potential *Sticky Keys* users will have previously been exposed to the default method, and would occasionally revert to it out of habit. Automatic deactivation of the utility

when this happens could cause major difficulties for new users, and this feature of the interface requires further investigation, and possibly redesign.

The sophistication of the *Sticky Keys* utility also caused difficulties for some participants when they activated more advanced functionality accidentally while using the basic level. While the positive reactions of many of the participants indicate that *Sticky Keys* has great potential, the participants in this experiment were supported in recovering from errors in their use of *Sticky Keys*. Future studies should examine the usability of the utility as a whole more closely, particularly the automatic deactivation feature. Similar studies evaluating the usability of all the existing configuration facilities are also recommended.

The results presented here imply that training in the use of *Sticky Keys* would be helpful for many potential users.

9.2.4 Use of Configuration Facilities

In Chapter 6, it was shown that while some of the keyboard access facilities available in standard operating systems are found to be useful and are actively made use of by their target population, they are not used by all those who could benefit from them. Other facilities are not widely used. In this study, the major barrier to their adoption was a lack of awareness of their existence. Users may not be able to discover or operate access features themselves, particularly when new to computing.

Lack of awareness, however, is not the only barrier. The process of adjusting the configuration of a computer before each session, and resetting it at the end can be perceived as error-prone, time consuming and not worth the effort, particularly where machines are shared by users with different requirements.

Some commonly observed difficulties presented in Chapter 4 are not addressed by existing software facilities. Many additional key errors and remote key errors may not be effectively eliminated by *Slow Keys*, for example. Missing key errors are also not tackled. There are opportunities for exploration of new utilities.

An example is the prototype of *Overlap Keys* presented in Chapter 6. *Overlap Keys* examines overlapping keystrokes on adjacent keys, and treats them as additional key errors. The prototype itself was primitive, deleting both characters rather than

choosing the most reasonable of the two, and the participants in the evaluation did not match the target user group for *Overlap Keys*, so the initial evaluation results were mixed. Nevertheless, the experiment did show that such a utility could operate unintrusively to correct additional key errors. Some participants expressed an interest in using such a utility, if its error corrections were reasonably accurate.

9.3 Modelling Keyboard Difficulties

Of the configuration facilities available, the majority are concerned with aspects of keyboard response. In the study described in Chapter 4 it was found that, for a human, keyboard difficulties are easier to identify from a user's inputs than mouse difficulties are. It is therefore easier to design mechanisms for automatic recognition of keyboard difficulties. Because of this, and the wide variety of configuration options available for keyboards, keyboard configuration was chosen as the domain to be modelled.

A model of keyboard skills and configuration requirements has been developed and presented, using the typing data from the study described in Chapter 4 to provide a sound empirical basis. The model is intended to provide a basis for a configuration support tool, as introduced in Chapter 3. Such a tool could have a number of different forms - two are outlined in Sections 9.5 and 9.6. In order to be as general as possible, the model does not rely on explicit user testing or questioning. Instead it bases recommendations on observation of the user typing English text, a task which is expected to occur naturally during computer use. This approach also has the advantage of reducing the load on the user, which is particularly important where users with disabilities are concerned.

Because the model has only minimal information about the input being used to assess each user's requirements, and the input data available is simple and reliable, existing symbolic user modelling techniques were inappropriate. Instead, statistically based, task-specific methods were used.

The resulting model addresses four areas of keyboard difficulty: use of modifier keys, long key press errors, bounce errors and additional key errors. In addition to recognising users prone to each of these difficulties, it makes recommendations for settings of the appropriate existing configuration facilities: *Sticky Keys*, *Repeat Keys* and *Bounce Keys*. While there are other keyboard configuration facilities which could

be modelled, these were the most common difficulties observed in Chapter 4 for which support existed. The use of *Slow Keys* to tackle additional key errors was not examined, since the errors of this type observed in Chapter 4 did not match the type of errors for which *Slow Keys* was designed. The observed errors were key presses as long as normal key presses, while *Slow Keys* is intended to eliminate brief key flicks.

9.3.1 Internal Evaluation

The development of the model was guided by internal evaluation using the recorded typing data described in Chapter 4. Most importantly, this process provided the basis for the choice of values for internal parameters. The completed model recommended key repeat delays that would have reduced the number of long key press errors from 2610 to 151. Use of *Sticky Keys* where recommended could have eliminated 54 of the 56 dropping errors, and would have helped nine of the eleven participants who reported difficulty in using modifier keys. All three participants prone to additional key errors were recognised, as were four of the seven who made bounce errors. Importantly, the model only once recommended an inappropriate facility for a user, and this recommendation was prompted by the user's misunderstanding of the use of modifier keys.

9.3.2 Evaluation Methodology

External evaluation of the model provided information about its generalisability to other users and different text passages. It also examined the sensitivity and stability of the model. Evaluation took the form of an empirical study involving twenty participants with motor disabilities and ten with no motor disability. Participants typed a given passage while the model examined their input. They then tried out configuration options as suggested by the model. The effectiveness of the model's recommendations was examined primarily in terms of error numbers and reported user satisfaction.

A number of limiting factors affected the evaluation methodology. For example, because of the slow typing speeds of many participants and the number of configuration options to be examined, it was not possible for every participant to try

every configuration option. It was also not possible to control properly for the effects of fatigue and practice. Relatively short passages of typing were examined, limiting the information available on model stability and sensitivity to changes in the user's input. Potential interference between different options was also not examined. While the experimental conditions themselves may have affected the speed and accuracy of some participants' typing, either positively or negatively, this does not invalidate the effectiveness with which the model identified the difficulties presented. However, the artificiality of the task may have affected the results. Participants were copying text, and did not make many changes to the text. A text creation task would have involved more editing, and the effect of this on the model is discussed in Section 9.4.1. Further studies would be necessary to remedy all of these limitations.

9.3.3 Results

The model's key repeat delay recommendations reduced the average error rate from 3.33% to 0.27 % over the twenty-three participants who tried an altered delay. All three of those with initial error rates over 10% had their error rates reduced to less than 0.3%. The highest error rates while using the recommended delay (up to 2.06%) were observed for those for whom the setting used was exactly that recommended by the model - the model's recommendations could be increased by a small constant, say one or two ticks, to reduce these observed error rates.

The model was quick to settle on a reasonable recommendation, typically stabilising within twenty keystrokes. The stability of the model was also good. Some participants changed their key press lengths over time, and the model was able to respond to these changes.

The recommendations made by the model on the use of *Sticky Keys* correlated well with those of the participants themselves, both before and after having tried the facility. Nevertheless, in nine cases the model failed to recognise potential *Sticky Keys* users. Only one of these found *Sticky Keys* essential, and all of them could use both hands for modifier key presses. In this aspect of configuration, user questioning or human observation would probably provide more accurate results.

The stability of this aspect of the model was generally good - changes of recommendation were rare. The number of uses of modifier keys made by the

participants before the final recommendation was reached was small. The model could therefore identify those participants who made errors in the use of modifier keys fairly quickly, depending on the number of modified characters in the text they were typing. However, there was some evidence of accumulation of evidence values over time, where they should ideally remain constant unless a user's typing changes. This may have reduced the quality of the results over a longer period - an increase in the evidence decay rate would address this problem.

Due to the low bounce error rates observed during evaluation, the model typically observed a long period of typing before ascertaining a need for *Bounce Keys*, in comparison with the period required for *Repeat Keys* or *Sticky Keys*. The model recommended the use of *Bounce Keys* consistently for the four participants with the highest bounce error rates - over 0.2 errors per 100 characters typed.

The accuracy and stability of the *Bounce Keys* recommendations made by the model were good. The threshold was effective in eliminating noise due to inaccuracies in discrimination between bounce errors and deliberately typed double letters.

The strong correlation (Spearman Rho = 0.947, $p<0.01$) between additional key error rates and the model's count of additional key errors shows that despite the uncertainty associated with classification of overlapping adjacent keystrokes the model is accurate enough to be useful in identifying those users who are prone to such errors. Together with the deliberate overlap rates, this information could be used to recommend a utility like *Overlap Keys*.

Nine of those who provided data for model development also took part in model evaluation. This overlap is likely to have exerted a positive bias on the results. Nevertheless, the model was shown to be effective for the twenty-one new participants. In particular, three novice touch typists were included in the evaluation, while none participated in the original study. The model should also be evaluated by expert touch typists, in order to ensure that no characteristics of fast touch typing are mistaken for input difficulties.

The model's recommendations showed significant correlation with those of the participants who had tried *Sticky Keys* and *Repeat Keys*. The possibility of participant bias has been discussed in Section 9.2.1.

The model evaluation showed that, for the copy typing task examined, the model was successful in identifying users with difficulties in all four of the areas studied, the only

exception being those who found the use of modifier keys awkward but achievable. Conversely, the model never recommended a facility for a user who had not exhibited the relevant difficulties. Where testable, the model's specific configuration recommendations were also accurate, with only a very few fine tuning alterations suggested by the evaluation. The stability of the model was also good, and again only minor modifications were suggested. Given these positive results, the reliable identification of configuration requirements from typing appears to be a feasible proposition, at least in the context of the aspects of keyboard use studied here.

9.4 Extending and Improving the Model

In addition to the minor modifications suggested by the evaluation results and mentioned previously, there are several notable ways in which the model could be extended. This section describes some improvements that could be made to the model to overcome existing limitations. It also outlines extensions of the model to other aspects of keyboard use, and to the use of pointing devices, and discusses alternative implementation techniques that may prove fruitful.

9.4.1 Modifications to the Existing Model

A number of potential improvements are described below.

- *Relax assumptions about the task.*

  Both the implementation and the evaluation of the model have been based on the assumption that a word processing application is being used, and the input is a continuous stream of text. While it is possible to base the model on languages other than English, including command or programming languages, the basic assumption of a continuous stream of input remains.

  This is a limiting assumption: in unconstrained word processing, users also spend time editing existing text, and moving around documents. While some sequences of input characters may form whole words and sentences, others will be single letters, unrelated to the letters typed previously. For example, a user may type a

'q', then use the mouse to point and click on a different location, then type another 'q'. While the timing of the two 'q's would make it unlikely that they would be considered a bounce error, they may be treated as a deliberate double letter, and the timing information would be incorporated into the current model of how quickly the user typically types double letters. As another example, suppose the user pressed and released a modifier key, then pointed and clicked with the mouse, then typed a letter. The current version of the model would consider the sequence a dropping error, whereas the intervening use of the mouse strongly suggests that the user had actually abandoned an intended *Shift* operation in favour of inserting a character.

Extensions to the model would be required in order to ensure that input from free editing could be correctly interpreted. This would include taking mouse movement into account. The effect of heavy use of the arrow keys on the model should also be examined.

- *Handle changes of user.*

  If the model is to be useful on shared machines, where there may be no explicit indication of changes in user, it needs to respond quickly when there is a sudden change in the typing characteristics being observed, since the current recommended configuration may no longer be appropriate. The typing data observed suggests that there are significant differences between the typing styles of different users with motor disabilities. Other researchers have also found significant differences between individuals in the timing of keystrokes for non-disabled keyboard users (Gentner, 1983; Legget et al., 1991; Mahar et al., 1995). Currently, the only aspect of the model that responds quickly to sudden changes in typing characteristics is the key repeat delay chosen. Ideally, there should be a mechanism for identifying changes in user, perhaps based on that used in the key repeat delay calculations. When such a change is identified, the current user model could be discarded and a new one initialised.

- *Match the form of the recommendations to the choices available for the configuration facilities.*

  The model currently makes specific recommendations, in terms of ticks, for values such as the key repeat delay and bounce delay. However, on the majority of

systems users are presented with a qualitative, rather than quantitative scale of choices. (An exception is Windows 95, which offers values in seconds for some options, and qualitative values for others, as described in Chapter 2.) It would currently be difficult for users to translate the model's recommendations into a specific choice of setting. A more usable interface would present recommendations in the same terms as they appeared in the interface being used.

- *Improve identification of users with difficulties.*

  The model currently makes configuration recommendations on the basis of observed performance errors and, in the case of *Sticky Keys*, input patterns indicative of difficulty. As has been observed previously, it cannot identify users who have difficulty, but avoid making errors at the expense of effort, and perhaps also time. This phenomenon may be relevant to all areas of the model, not just *Sticky Keys*. It is possible that there may be input cues which would allow identification of such users. For example, timing information could be exploited. Further examination of the recorded log data would be required. There is however, no guarantee that such cues exist. For example, how is the system to differentiate between the many possible reasons for a slow input rate? The inability to diagnose the effort required of a user may be a hard limitation of this approach to configuration.

All of these suggested extensions would require thorough evaluation with users with and without motor disabilities. Ideally, further evaluation would also involve expert touch typists, and other classes of keyboard user not strongly represented in the current studies, such as children, occasional users and novices. When accommodating unconstrained word processing activities, as opposed to text copying, further evaluation should involve as natural and unconstrained a set of tasks as possible. Longer periods of evaluation would also be useful, in order to increase confidence in the stability of the model, and it's flexibility in responding to changes in and between users. Evaluation with different users sharing a machine would be required in the case of extensions designed to recognise changes in user. In addition, the results for *Bounce Keys* have not yet been fully evaluated - for this, a non-Macintosh platform would be required.

Due to the methodological restrictions imposed by empirical work with people, particularly those who may find using a keyboard tiring and difficult, some of these evaluation goals may not be easy to achieve.

9.4.2 Modelling Other Errors

As described in Chapter 2, the available keyboard configuration facilities and the options they present vary between platforms. The current version of the model does not make recommendations for all of the available options on any platform. For example, no recommendation for the key repeat rate is made, *Slow Keys* is ignored entirely (due to lack of relevant data in the initial study), and the model does not indicate whether the automatic disabling of *Sticky Keys* should be used. Ideally, the model would be extended to cover the whole range of options available in the keyboard configuration facilities of modern platforms. This would require further empirical data gathering, study of users' preferences, implementation of model extensions covering the omitted features, and evaluation. In practice, it may not be possible to choose appropriate settings for some features without consulting the user. Further research is required to establish the best mechanism for individual options.

Modelling of mouse configuration requirements, and cursor control skills in general, would also be worthwhile. On the surface, this appears to be a more difficult problem than modelling keyboard use, partly because correct and erroneous inputs are less easily distinguishable, and partly because the relationship between input patterns and appropriate configurations is not well defined.

Nevertheless, characteristics of pointing device usage could be interpretable. Barelle et al. (1996) have studied the use of pointing characteristics in user identity verification. They developed an algorithm which could discriminate between ten different users of a cordless pressure pen with 62% accuracy given a single pointing movement. Given ten pointing movements, accuracy improved to 88%. Similar results were observed when the participants used a puck (cordless cursor) as the pointing device. These results indicate that individual differences in pointing device usage are significant, even among non-disabled users.

Users with motor disabilities would be likely to show even larger individual differences. It may be that some of these differences are indicative of specific

difficulties in using the pointing device. For example, users who find it difficult to lift a mouse and replace it on the table may produce recognisable patterns in the attempt. Recognition of such patterns, if they exist, would allow simple configuration options to be suggested. As there are few software configuration options available for mice, some of these may be suggestions for external aids. A trackerball or joystick could be suggested as an alternative to the mouse, for example. Further investigation of specific mouse difficulties, and automatic recognition of these difficulties, may help to spur research into new software configuration facilities for pointing devices.

9.4.3 Alternative Approaches

Other statistical modelling techniques such as nearest neighbour classification algorithms, or connectionist techniques, have not been considered in this thesis, but may also be suitable for modelling users' input abilities. Much user modelling is essentially a classification task (Allen, 1990; Finlay, 1990), and classification of input patterns is an area in which connectionist and other statistical techniques can perform very well. Being example based rather than knowledge based, they do not require knowledge to be made explicit, which could make these techniques particularly useful in the recognition of mouse difficulties.

Statistical approaches have been successful in the identification of users from their typing characteristics (Legget et al., 1991; Napier et al., 1995). The primary drawback of such approaches is that they typically require a set of initial examples, which are used as a reference point in the classification of further cases. For example, Napier et al. required each subject to type a 1000 character reference text. Before applying similar methods to the recognition of keyboard and mouse difficulties, a significant volume of reference data exemplifying the difficulties would be required.

Connectionist techniques have also been applied to the recognition of users from their typing characteristics. Brown and Rogers (1993) based their approach on multi-layer perceptrons (Rumelhart and McClelland, 1986), which can be trained using a set of examples, and are then able to generalise from this training to classify previously unseen examples. However, Brown and Rogers used pre-processing to remove noise from their data before training the net - greater tolerance to noise would be necessary to enable their technique to be used dynamically.

One famous application of multi-layer perceptrons is NetTalk (Sejnowski and Rosenberg, 1987), a network which learned to translate English text into phonemes. The net operated using a window of seven characters, which was scanned over the text. The net itself was a three-layer architecture, trained using the backpropagation algorithm (Rumelhart, Hinton and Williams, 1986). NetTalk is interesting in that it processes a stream of keyboard input, to produce a stream of output. Because the essential features defining the correct phoneme are locally placed in the input stream, analysis of a small window is sufficient to allow a good standard of recognition. Similarly, the characteristics defining keyboard performance errors are also local features of the input stream, and could perhaps be detected in a similar way.

Because timing information is important in performance error recognition, recurrent networks may be appropriate. These are networks which have an additional feedback loop, so that the output at a given time depends not only on the current input but also on the output from the previous step. Recurrent networks have been applied to problems such as the recognition of sign language gestures (Murakami and Taguchi, 1991) with some success. This technique may be transferable to the interpretation of mouse movements. The greatest problem with recognising patterns in continuous movement is that of knowing where to segment the movement stream so that analysis of the joins between meaningful movements is not attempted. Murakami and Taguchi sidestepped this difficulty by requiring movements to begin and end in a fixed position. Analysis of mouse movements for stereotypical patterns may be an easier problem, since the majority of mouse movements have (or are intended to have) a well-defined ending, in a mouse click.

Associative memories, in which inputs are used as keys to evoke specific output responses, have been applied to spelling correction, which can be thought of as a complementary application to error recognition. In an extensive review, Cherkassky et al, (1992) investigated several neural and conventional architectures for spelling correction, and concluded that a supervised learning associative memory was the most promising neural network approach. The net is trained on correct spellings, and then when a misspelling is presented, the nearest word on which it has been trained is returned as the output. Beale and Finlay (1989) used an associative network to distinguish between expert and novice users of a functional interactive language. They found that the results were more accurate than those of a similar study which used an adaptive learning technique. In a second study (Beale and Finlay, 1989) they applied the same technique to the classification of tasks in a bibliographic data base. This

application was used interactively, and the results highlight a similar problem to those of the gesture recognition research - when a continual stream of input is presented the net attempts to classify the joins between tasks, as well as the tasks themselves. This produced a success rate of 56%, while removal of classifications of joins from the results gave an 87% success rate.

In the domain considered by this thesis, users would be classified with respect to the available keyboard and mouse configuration options, and also their general keyboard and mouse skills. The input to this classification process would be a representation of the keystrokes and/or mouse movements made by the user. For some of the configuration options, such as *Sticky Keys*, the classification would be binary - the facility is either required or it is not. For others, such as the key repeat delay, users would be placed on a scale of possible settings. In measuring general keyboard and mouse skills, the classification would indicate, for example, levels of skill at dragging the mouse, or homing in accurately on a specific key.

It would be interesting to apply connectionist techniques to the recognition of both keyboard and mouse difficulties, and to compare the results with those of symbolic techniques. Different techniques could be used for different aspects of the problem. The data gathered in the two empirical studies could be used as initial training data. Ultimately, a larger volume of data would be necessary. Given the severe restrictions on the volume of data that can be recorded in a single session for many of the participants in the studies described here, gathering such data could be time consuming.

The success of NetTalk shows that in processing character based input it is not always necessary to segment the input stream, but the introduction of timing information in some form would undoubtedly be necessary. One of the reasons why the recognition of mouse difficulties has not been tackled in this thesis is the difficulty in formulating appropriate algorithms in the absence of specific task knowledge. Neural network approaches offer a potential means to circumvent this difficulty. However, obstacles such as the segmentation of continuous mouse input, and the incorporation of timing information would provide substantial challenges for this approach.

9.5 Applications I - Assessment

There are several ways in which a model of keyboard skills could be used to support keyboard configuration. This section discusses one simple approach - that of using the model explicitly as a mechanism for assessing a user's configuration requirements.

In Chapter 2, Section 2.9, the assessment process was introduced. During assessment, users with disabilities and clinicians choose the most appropriate input devices and configure them to suit the user. Often, this involves a process of trial and error, both with different devices and with different configurations. This can be time consuming and tiring. Having chosen an input device and an initial configuration, follow-up training and support are necessary to ensure that the configuration remains appropriate as the user becomes more familiar with the device. Unfortunately, these services are often limited.

Keyboard configuration is usually performed by observation of the user typing at the keyboard (Lee and Thomas, 1990). Sometimes, however, it can be difficult to differentiate between different errors by observation alone. For example, bounce errors and doubling errors produce the same output, but have very different configuration solutions. Even users themselves cannot always pinpoint the cause of an error, particularly novice users unfamiliar with keyboards.

A model of keyboard skills such as the one presented in this thesis could make a useful contribution to the assessment process. It could help to make configuration decisions, while the user and clinician focus on other aspects of keyboard use such as the user's positioning and comfort, the ease with which they can reach all the keys on the keyboard, or the need for external aids such as arm rests, wrist rests or keyguards. In addition, the model could be used directly by the user outside of the assessment period. Changes in user requirements could be reported to the user by the model, and then implemented by the model under the control of the user.

Similarly, the model could be used by individual keyboard users who have not undergone a formal assessment process. An application incorporating the model could record typing samples for the user, introduce the relevant configuration facilities, suggest appropriate settings, allow the user a chance to try out the facilities and settings suggested, and implement configuration changes requested by the user. It could also allow the user to explore other configuration facilities which may be of interest. Periodical activation of this application would allow the user to keep their

configuration well matched to changing requirements. The application could also store configuration settings for specific users, allowing quick and easy configuration changes for machines used by several people. This latter function is similar to the user profiles that can be stored in Microsoft Windows 95. Currently, Macintosh machines do not offer such a facility.

Automated support for configuration of this kind could help users who have limited access to expert human advice to perform their own configuration. With an appropriate interface, it could increase the independence of computer users with motor disabilities, particularly novice users. It would reduce the difficulty of finding out about the various options available - a user would simply need to know about the support facility, and could find out about all configuration options through this application. It would not, however, solve all problems of lack of knowledge, since the user would be required to find out about the support application itself. The following section describes an alternative application of the model which has the potential to reduce or eliminate this difficulty.

9.6 Applications II - Dynamic Configuration

A more ambitious application of the model would be to allow it to draw conclusions about the current user automatically as they go about their normal tasks. Suggestions could then be presented to the user in some way, or configuration changes could be made unobtrusively. This approach has been introduced in Chapter 3, Section 3.3.

As more access facilities are developed, the large range of options available will exacerbate the problem of informing users of facilities they may wish to use. A configuration support tool could identify a subset of the available facilities that are relevant to the current user, and focus the interaction on that set.

Active configuration has a number of attractions beyond simply making users aware of facilities they may be interested in. It may also present a solution to the difficulties observed where computers are shared by many different users, given the perceived difficulty of adjusting the access settings for each user. An active configuration utility could help to ease the transition between users, allowing them to configure shared machines without the help of a teacher or system administrator.

The interface to such a system would need to be very carefully designed in order to be acceptable to users, given the negative reactions to some of the adaptive systems described in Chapter 3. There are a number of ways in which the system could react when a configuration need is identified, each with its own advantages and disadvantages. Different approaches may be necessary for different configuration facilities. The major options are:

- *Present the suggestion immediately.* As the user is working, their errors might suggest a change on the keyboard configuration. The system could interrupt the user in order to present this suggestion. If, as is often the case, the user is aware of and frustrated by the errors occurring, this may be the best course of action. Immediate alteration to the configuration could remove the source of the errors before the user becomes too frustrated and abandons their task. It does not require the user to have any prior knowledge of configuration mechanisms, or the configuration system itself. The user will also be aware of, and in control of, the configuration change, so that the change in behaviour of the system will be predictable and understandable.

  On the other hand, if the user is concentrating on what they are typing, interrupting them with configuration suggestions is likely to be intrusive and irritating. This could cause users to reject the system and ignore its suggestions.

- *Signal the availability of a suggestion.* A less intrusive option would be to signal the presence of a suggestion by some form of screen icon or sound. Users could then choose when to view the suggestion. This reduces the risk of rejection of the system, but introduces the possibility that users will not notice that a suggestion is being made. It requires users to be aware that the system may make suggestions, and they must also know how to access these suggestions. This reintroduces the problem of lack of user knowledge. However, the knowledge required would be less than the knowledge required to understand and use all the available configuration facilities, so this may be an acceptable compromise.

- *Store the suggestion until the user requests it.* A third option is to make no active suggestions, but to store them until the user requests support with configuration. This places the user entirely in control, and eliminates the risk of distracting them from their task. Again, this would require the user to know how to request help,

but such a system could be built into an existing word processor help system quite naturally, making it easier to discover.

- *Implement the suggestion without interrupting the user.* Finally, for some facilities it would be possible to activate or deactivate them, or change the current settings, without waiting for the user to make decisions. In the best case, errors would simply disappear as the user continued to type. For some facilities, users may remain entirely unaware of the underlying configuration change. For users who are already overloaded with new information about a complex system, this may be the ideal option.

  In the worst case, this could lead to the user becoming confused by the apparently independent and unpredictable changes in the system. This in turn could lead to rejection of the whole machine, not just the adaptive system causing the changes. A compromise might be to make automatic changes, but alert the user to the existence of those changes by providing some on-screen or auditory cue which they could follow up to determine what had been changed and why, and to reverse unwanted changes. As with the second option, this requires the user to know that changes are possible, and how to find out about them.

  For facilities which require the user to alter the way in which they use the keyboard, such as *Sticky Keys*, this option is not feasible. For *Repeat Keys*, where adjustments of the repeat delay short of infinite delays have little or no effect on the way in which the keyboard is used, it may be ideal.

Further research is necessary to investigate appropriate adaption mechanisms for each configuration option. For each facility, user reactions to different strategies could be investigated, to see if a particular approach is more acceptable than others. If generally acceptable strategies were found, this could lead to the development of a dynamic configuration support system capable of overcoming many of the existing barriers to configuration and efficient keyboard use currently experienced by people with motor disabilities.

9.7 Conclusion

To summarise, the primary contributions of the research described in this thesis are:

1. Demonstration of a model identifying user difficulties for four of the major sources of keyboard error.
2. Application of user modelling to a new domain, in which traditional techniques were not appropriate.
3. Data describing the extent and nature of input difficulties experienced by twenty users with a variety of different motor disabilities when using keyboards and mice.
4. A methodology for empirical research on input device usage by users with motor disabilities.
5. The *InputLogger* tool for recording detailed input event logs on Macintosh platforms.
6. A preliminary assessment of the coverage of the existing configuration facilities with respect to the errors observed.
7. Preliminary examination of the usability of *Sticky Keys* and *Repeat Keys*, leading to suggestions for improvements to the interfaces.
8. Identification of current barriers to the usage of configuration facilities, the primary barrier being lack of awareness of the available options.
9. A proposal for a new configuration facility handling additional key errors - *Overlap Keys*.

Thus, the primary contribution of this thesis has been to demonstrate that simple user modelling techniques can successfully be used to model keyboarding difficulties. This has implications for the fields of assistive technology and human-computer interaction, since it suggests that automated, individualised configuration support is possible. This would be a potentially fruitful area for future research. Extension of the model to other areas of configuration, particularly mouse configuration, would also be worthwhile and challenging.

The model presented in this thesis is also of interest to user modelling practitioners, particularly those interested in developing generic user modelling tools, as an example of a new modelling domain.

The data gathered in the course of this research would be of benefit to developers of configuration facilities and other assistive software and hardware. They provide an indication of some of the frequently encountered difficulties, some of which are not addressed by existing configuration facilities. Similarly, the investigations into the use of existing facilities, and suggestions for new facilities may contribute to the development of new and better support tools for keyboard users with motor disabilities.

References

Aeberhard, J. (1998). Labour's new deal. *Ability*, **23**, 8-9. The magazine of the British Computer Society Disability Group.

Allen, R. (1990). User models: Theory, method and practice. *International Journal of Man-Machine Studies*, **32**, 511-543.

Alm, N., Arnott, J. L., and Newell, A. F. (1992). Prediction and conversational momentum in an augmentative communication system. *Communications of the A.C.M.*, **35**(5), 46--57.

Ambrosini, L., Cirillo, V., and Micarelli, A. (1997). A hybrid architecture for user-adapted information filtering on the world wide web. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 59-61, New York. Springer Wein.

Anderson, T. and Smith, C. (1996). 'Composability': widening participation in music making for people with disabilities via music software and controller solutions. In *Proceedings of the Second Annual ACM Conference on Assistive Technologies*, pages 110-116, USA. ACM. ACM ISBN: 0-89791-776-6.

Anon. (1952). Parkinsonism. In *Disabilities and how to live with them*, pages 52-58. The Lancet Limited, 7 Adam Street, Adelphi, London.

August, S. and Weiss, P. (1992). A human-factors approach to adapted access device prescription and customization. *Journal of Rehabilitation Research and Development*, **29**(4), 64-77.

Baecker, R., Grudin, J., Buxton, W., and Greenberg, S., editors (1995). *Readings in Human-Computer Interaction: Toward the Year 2000*. Morgan Kaufmann Publishers, Inc, San Francisco, CA, 2nd edition.

Baecker, R. M. and Buxton, W. A. S. (1987). *Readings in Human-Computer Interaction: A Multidisciplinary Approach*. Morgan Kaufmann, Los Altos, CA.

Bailey, R. W. (1983). *Human Error in Computer Systems*. Prentice-Hall Inc., Englewood Cliffs, NJ. ISBN 0-13-445056-6.

Barrelle, K., Laverty, W., Henderson, R., Gough, J., Wagner, M., and Hiron, M. (1996). User verification through pointing characteristics: an exploration examination. *International Journal of Human-Computer Studies*, **45**(1), 47-57.

Bauer, M. (1996). A Dempster-Schafer approach to modeling agent preferences for plan recognition. *User Modeling and User-Adapted Interaction*, **5**, 317-348.

Beale, R. and Finlay, J. (1989). User modelling with a neural system. YCS 118, University of York, Department of Computer Science, Heslington, York.

Beck, J., Stern, M., and Woolf, B. P. (1997). Using the student model to control problem difficulty. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 277-288, New York. Springer Wein.

Benyon, D. (1985). MONITOR. A self-adaptive user interface. In B. Shackel, editor, *Human-Computer Interaction - INTERACT '84*, pages 335-341, North-Holland. Elsevier Science Publishers B.V.

Benyon, D. and Murray, D. (1993a). Adaptive systems: from intelligent tutoring to autonomous agents. *Knowledge-based Systems*, **6**(4), 197-219.

Benyon, D. and Murray, D. (1993b). Applying user modelling to human-computer interaction design. *Artificial Intelligence Review*, **7**, 199-226.

Bewley, W., Roberts, T., Schroit, D., and Verplank, W. (1990). Human factors testing in the design of Xerox's 8010 'Star' office workstation. In J. Preece and L. Keller, editors, *Human-Computer Interaction: Selected Readings*. Prentice-Hall, Hemel Hempstead.

Borden, P. (1991). *User's Guide for AccessDOS*. The Trace Research and Development Center, Madison, WI, 1.0 edition.

Brajnik, G. and Tasso, C. (1994). A shell for developing non-monotonic user modeling systems. *International Journal of Human-Computer Studies*, **40**, 31-62.

Brewster, S., Raty, V.-P., and Kortekangas, A. (1996). Enhancing scanning input with non-speech sounds. In *Proceedings of the Second Annual ACM Conference on Assistive Technologies*, pages 10-14, USA. ACM. ACM ISBN: 0-89791-776-6.

Broadbent, S. and Curran, S. (1992). *The Assessment, Disability and Technology Handbook*. North West Regional ACCESS Centre and Oldham Education Department, Oldham.

Brown, C. (1992). Assistive technology computers and people with disabilities. *Communications of the A.C.M.*, **35**(5), 36-45.

Brown, J. and Burton, R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science*, **2**, 155-192.

Brown, M. and Rogers, S. (1993). User identification via keystroke characteristics of typed names using neural networks. *International Journal of Man-Machine Studies*, **39**(6), 999-1014.

Browne, D. (1990). Conclusions. In D. Browne, M. Norman, and D. Riches, editors, *Adaptive User Interfaces*, pages 195-212. Academic Press, London.

Brownlow, N., Shein, F., Thomas, D., Milner, M., and Parnes, P. (1989). Direct manipulation: Problems with pointing devices. In *Resna '89: Proceedings of the 12th Annual Conference*, pages 246-247, Washington D.C. Resna Press.

Brusilowski, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction*, **6**(2-3), 87-129.

Bull, S. (1997). See yourself write: A simple student model to make students think. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 315-326, New York. Springer Wein.

Carberry, S. (1990). Incorporating default inferences into plan recognition. In *AAAI 90: Proceedings of the Eighth National Conference of the American Association for Artificial Intelligence, Boston, MA*, volume 1, pages 471-478, Menlo Park, CA. AAAI Press.

Card, S., Moran, T., and Newell, A. (1987). The keystroke-level model for user performance time with interactive systems. In R. M. Baecker and W. A. S. Buxton, editors, *Readings in Human-Computer Interaction: A Multidisciplinary Approach*, chapter 5, pages 192-206. Morgan Kaufmann, Los Altos, Calif.

Casali, S. (1995). A physical skills based strategy for choosing an appropriate interface method. In A. D. N. Edwards, editor, *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*, chapter 17, pages 315-341. Cambridge University Press.

Cherkassky, V., Vassilas, N., Brodt, G., and Wechsler, H. (1992). Conventional and associative memory approaches to automatic spelling correction. *Engineering Applications of Artificial Intelligence*, **5**(3), 223-237.

Chizinsky, K. (1990). Keyboard access to the Apple Macintosh. In *Resna '90: Proceedings of the 13th Annual Conference*, pages 253-254, Washington D.C. Resna Press.

Clancey, W. (1987). *Knowledge-Based Tutoring: The GUIDON Program*. MIT Press.

Cooper, W. (1983). *Cognitive Aspects of Skilled Typewriting*. Springer-Verlag, New York.

Corbett, A. and Bhatnagar, A. (1997). Student modeling in the ACT programming tutor: Adjusting a procedural learning model with declarative knowledge. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 243-254, New York. Springer Wein.

Cox, A. (1996). Access enabled? In F. E. D. Agency, editor, *Creating Connections: College Innovations in Flexibility, Access and Participation*, chapter 2, pages 17-27. Association of Colleges, Bristol. ISBN 1853 38 4348.

Cypher, A. (1991). EAGER: Programming repetitive tasks by example. In *Proceedings of CHI*, pages 33-39, New York. ACM.

Damerau, F. (1964). A technique for computer detection and correction of spelling errors. *Communications of the A.C.M.*, **7**(3), 171-6.

Damper, R., Tranchant, M., and Lewis, S. (1996). Speech versus keying in command and control: effect of concurrent tasking. *International Journal of Human-Computer Studies*, **45**, 337-348.

DeCoste, D. (1997). Augmentative and alternative communication assessment strategies: Motor access and visual considerations. In S. Glennen and D. DeCoste, editors, *Handbook of Augmentative and Alternative Communication*, chapter 7, pages 243-282. Singular Publishing Group, Inc, San Diego.

Debevc, M., Meyer, B., Donlagic, D., and Svecko, R. (1996). Design and evaluation of an adaptive icon toolbar. *User Modeling and User-Adapted Interaction*, **6**(1), 1-21.

Demasco, P. W. and McCoy, K. F. (1992). Generating text from compressed input: An intelligent interface for people with severe motor impairments. *Communications of the A.C.M.*, **35**(5), 68-78.

Dieterich, H., Malinowski, U., Kühme, T., and Schneider-Hufschmidt, M. (1993). State of the art in adaptive user interfaces. In M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski, editors, *Adaptive User Interfaces: Principles and Practice*, volume 10 of *Human Factors in Information Technology*, pages 13-48. Elsevier Science Publishers B.V., Amsterdam.

Dillon, T. and Norcio, A. (1997). User performance and acceptance of a speech-input interface in a health assessment task. *International Journal of Human-Computer Studies*, **47**(4), 591-602.

Edwards, A. D. N. (1995). *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*. Cambridge Series on Human-Computer Interaction 7. Cambridge University Press.

Egan, D. E. (1988). Individual differences in human-computer interaction. In M. Helander, editor, *Handbook of Human-Computer Interaction*, chapter 24, pages 543-580. Elsevier Science Publishers B.V.

Finlay, J. (1990). *Modelling Users by Classification: An Example-Based Approach*. Ph.D. thesis, University of York.

Gentner, D., Grudin, J., Larochelle, S., Norman, D., and Rumelhart, D. (1983). A glossary of terms including a classification of typing errors. In W. E. Cooper, editor, *Cognitive Aspects of Skilled Typewriting*, chapter 2, pages 39-44. Springer-Verlag, New York.

Gerard, M., Jones, S., Smith, L., Thomas, R., and Wang, T. (1994). An ergonomic evaluation of the Kinesis Ergonomic Computer Keyboard. *Ergonomics*, **37**(10), 1661-1668.

Gillan, D., Holden, K., Adam, S., Rudisill, M., and Magee, L. (1990). How does Fitts' law fit pointing and dragging? In *Proceedings of CHI*, pages 227-234, New York. ACM.

Glennen, S. and DeCoste, D. (1997). *Handbook of Augmentative and Alternative Communication*. Singular Publishing Group, Inc, San Diego.

Glinert, E. P. and York, B. W. (1992). Computers and people with disabilities. *Communications of the A.C.M.*, **35**(5), 32-35.

Gordon, J. and Shortliffe, E. (1985). A method for managing evidential reasoning in a hierarchical hypothesis space. *Artificial Intelligence*, **26**, 323-357.

Greenberg, S. and Witten, I. (1985). Adaptive personalised interfaces: A question of viability. *Behaviour and Information Technology*, **4**(1), 31-45.

Greenstein, J. and Arnaut, L. (1987). Human factors aspects of manual computer input devices. In G. Salvendy, editor, *Handbook of Human Factors*, chapter 11.4, pages 1450-1489. John Wiley and Sons, New York.

Grudin, J. (1983). Error patterns in novice and skilled transcription typing. In W. E. Cooper, editor, *Cognitive Aspects of Skilled Typewriting*, chapter 6, pages 121-139. Springer-Verlag, New York.

Grudin, J. (1989). The case against user interface consistency. *Communications of the A.C.M.*, **32**(10), 1164-1173.

Gutkauf, B., Thies, S., and Domik, G. (1997). A user-adaptive chart editing system based on user modeling and critiquing. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 159-170, New York. Springer Wein.

Hansen, P. K. and Wanner, J. (1993). Software drivers for pointers used by persons with disabilities. In *Resna '93: Proceedings of the 16th Annual Conference*, pages 443-445, Washington D.C. Resna Press.

Hargreaves, W., Rempel, D., Halpern, N., Markison, R., Kroemer, K., and Litewka, J. (1992). Toward a more humane keyboard. In *Proceedings of CHI*, pages 365-368, New York. ACM.

Hockley, A. (1986). Adaptive user interfaces for information systems: an evaluation. In P. Zunde and J. Agrawal, editors, *Empirical Foundations of Information and Software Science IV: Empirical Methods of Evaluation of Man-Machine Interfaces*, pages 149-161. Plenum Press, New York.

Holt, P., Dubs, S., Jones, M., and Greer, J. (1991). The state of student modelling. In J. Greer and G. McCalla, editors, *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, NATO ASI Series F: Computer and Systems Sciences, Vol. 125, pages 3-35. Springer-Verlag, Berlin.

Horstmann, H. and Levine, S. (1991). The effectiveness of word prediction. In *Resna '91: Proceedings of the 14th Annual Conference*, pages 100-102, Washington D.C. Resna Press.

Horstmann Koester, H. and Levine, S. (1994). Learning and performance of able-bodied individuals using scanning systems with and without word prediction. *Assistive Technology*, **6**(1), 42-53.

Horvitz, E. (1997). Agents with beliefs: Reflections on Bayesian methods for user modeling. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 441-442, New York. Springer Wein.

Hurlburt, M. and Ottenbacher, K. (1992). An examination of direct selection typing rate and accuracy for persons with high-level spinal-cord injury using QWERTY and default on-screen keyboards. *Journal of Rehabilitation Research and Development*, **29**(4), 54-63.

Innocent, P. R. (1982). Towards self-adaptive interface systems. *International Journal of Man-Machine Studies*, **16**, 287-299.

Jacob, R. (1996). Human-computer interaction: Input devices. *ACM Computing Surveys*, **28**(1), 177-179.

Jameson, A. (1996). Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction*, **5**, 193-251.

Jameson, A., Paris, C., and Tasso, C. (1997). *User Modeling: Proceedings of the Sixth International Conference*. CISM Courses and Lectures No. 383. Springer Wein, New York.

Kass, R. and Finin, T. (1988). A general user modelling facility. In *Proceedings of Computer Human Interaction*, pages 145-150, New York. ACM.

Kass, R. and Finin, T. (1989). The role of user models in cooperative interactive systems. *International Journal of Intelligent Systems*, **4**, 81-112.

Kay, J. (1994). Lies, damned lies and stereotypes: Pragmatic approximations of users. In *Fourth International Conference on User Modeling: Proceedings of the conference*, pages 175-184. MITRE Corporation.

Kondraske, G. (1988). Rehabilitation engineering: Towards a systematic process. *IEEE Engineering in Medicine and Biology Magazine*, **7**(3), 11-15.

Kühme, T. (1993). User-centered approach to adaptive interfaces. *Knowledge-based Systems*, **6**(4), 239-248.

Kühme, T. and Schneider-Hufschmidt, M. (1993). Introduction. In M. Schneider-Hufschmidt, T. Kühme, and U. Malinowski, editors, *Adaptive User Interfaces: Principles and Practice*, volume 10 of *Human Factors in Information Technology*, pages 1-9. Elsevier Science Publishers B.V., Amsterdam.

Lazzaro, J. J. (1995). *Adapting PCs for Disabilities*. Addison Wesley.

Lee, C. (1989). Access to Microsoft Windows 2.0 for users with physical disabilities. In *Resna '89: Proceedings of the 12th Annual Conference*, pages 23-24, Washington D.C. Resna Press.

Lee, C. and Vanderheiden, G. (1987). Keyboard equivalent for mouse input. In *Resna '87: Proceedings of the 10th Annual Conference*, pages 711-713, Washington D.C. Resna Press.

Lee, K. S. and Thomas, D. J. (1990). *Control of Computer-Based Technology for People with Physical Disabilities: an Assessment Manual*. University of Toronto Press, Canada.

Legget, J., Williams, G., Usnick, M., and Longnecker, M. (1991). Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies*, **35**(6), 859-870.

Lerner, B. S. (1992). Automated customization of structure editors. *International Journal of Man-Machine Studies*, **37**(4), 529-563.

Linden, G., Hanks, S., and Lesh, N. (1997). Interactive assessment of user preference models: The automated travel assistant. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 67-78, New York. Springer Wein.

MacKenzie, S. and Buxton, W. (1994). Prediction of pointing and dragging times in graphical user interfaces. *Interacting With Computers*, **6**(2), 213-227.

MacKenzie, S., Sellen, A., and Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. In *Proceedings of CHI*, pages 161-166, New York. ACM.

Mahar, D., Napier, R., Wagner, M., Laverty, W., Henderson, R., and Hiron, M. (1995). Optimizing digraph-latency based biometric typist verification systems: inter and intra typist differences in digraph latency distributions. *International Journal of Human-Computer Studies*, **43**, 579-592.

Maskery, H. S. (1985). Adaptive interfaces for naive users - an experimental study. In B. Shackel, editor, *Human-Computer Interaction - INTERACT'84*, pages 343-349, Amsterdam. Elsevier Science Publishers (North-Holland).

Mason, M. (1986). Adaptive command prompting in an on-line documentation system. *International Journal of Man-Machine Studies*, **25**, 33-51.

McClelland, J. and Rumelhart, D. (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* MIT Press, Cambridge, MA.

McCoy, K. and Suri, L. (1996). English error correction: A syntactic user model based on principled "mal-rule" scoring. In *Proceedings of UM96: The Fifth International Conference on User Modeling*.

McMillan, W. W. (1992). Computing for users with special needs and models of computer-human interaction. In *Proceedings of CHI*, pages 143-148, New York. ACM.

McTear, M. (1993). User modelling for adaptive computer systems: a survey of recent developments. *Artificial Intelligence Review*, **7**, 157--184.

Metrowerks Inc. (1993). CodeWarrior 6. The Trimex Building, Route 11, Mooers, NY 12958, USA.

Meyer, B. (1994). Adaptive performance support: User acceptance of a self-adapting system. In *Fourth International Conference on User Modeling: Proceedings of the Conference*, pages 65-70, Bedford, MA. MITRE Corporation.

Michalski, R., Carbonell, J., and Mitchell, T. (1983). *Machine Learning: An Artificial Intelligence Approach*. Springer.

Millar, S. V. and Nisbet, P. D. (1993). *Accelerated Writing for People with Disabilities*. CALL Centre and Scottish Office Education Department, CALL Centre, University of Edinburgh. ISBN 1 898042 01 2.

Miller, D. and Swain, A. (1987). Human error and human reliability. In G. Salvendy, editor, *Handbook of Human Factors*, chapter 2.8, pages 219-250. John Wiley and Sons, New York.

Molnar, K. and Kletke, M. (1996). The impacts on user performance and satisfaction of a voice-based front-end interface for a software tool. *International Journal of Human-Computer Studies*, **45**(3), 287-303.

Morris, N., Rouse, W., and Ward, S. (1988). Studies of dynamic task allocation in an aerial search environment. *IEEE Transactions on Systems, Man, and Cybernetics*, **18**(2), 376-389.

Murakami, K. and Taguchi, H. (1991). Gesture recognition using recurrent neural networks. In *Proceedings of CHI*, pages 237-242, New York. ACM.

Napier, R., Laverty, W., Mahar, D., Henderson, R., Hiron, M., and Wagner, M. (1995). Keyboard user verification: toward an accurate, efficient, and ecologically valid algorithm. *International Journal of Human-Computer Studies*, **43**(2), 213--222.

Newell, A. (1995). Extra-ordinary human-computer interaction. In A. D. N. Edwards, editor, *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*, chapter 1, pages 3-18. Cambridge University Press.

Newell, A., Arnott, J., Cairns, A., Ricketts, I., and Gregor, P. (1995). Intelligent systems for speech and language impaired people: A portfolio of research. In A. D. N. Edwards, editor, *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*, chapter 5, pages 83-101. Cambridge University Press.

Nisbet, P. D. and Poon, P. (In press). *Special Access Technology*. CALL Centre and Scottish Office Education Department, CALL Centre, University of Edinburgh. ISBN 1 898042 11 X.

Norcio, A. and Stanley, J. (1989). Adaptive human-computer interaction: A literature survey and perspective. *IEEE Transactions on Systems, Man, and Cybernetics*, **19**(2), 399-408.

Norman, D. A. (1981). Categorization of action slips. *Psychological Review*, **88**(1), 1-15.

Norman, D. A. (1983). Design rules based on analyses of human error. *Communications of the A.C.M.*, **26**(4), 254-258.

Novak, M. and Vanderheiden, G. (1993). Extending the user interface for X windows to include persons with disabilities. In *Resna '93: Proceedings of the 16th Annual Conference*, pages 435-436, Washington D.C. Resna Press.

Novak, M., Schauer, J., Hinkens, J., and Vanderheiden, G. (1991). Providing computer access features under DOS. In *Resna '91: Proceedings of the 14th Annual Conference*, pages 163-165, Washington D.C. Resna Press.

Noyes, J. (1983). The QWERTY keyboard: a review. *International Journal of Man-Machine Studies*, **18**, 265-281.

Peterson, J. (1980). Computer programs for detecting and correcting spelling errors. *Communications of the A.C.M.*, **23**(12), 676-687.

Polson, M. and Richardson, J. (1988). *Foundations of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates, Inc, New Jersey.

Potosnak, K. M. (1988). Keys and Keyboards. In M. Helander, editor, *Handbook of HCI*, pages 475-494, North-Holland. Elsevier Science Publishers B.V.

Poulson, D., Ashby, M., and Richardson, S. (1996). *Userfit: A practical handbook on user-centred design for Assistive Technology*. European Commission, Brussels-Luxembourg. TIDE 1062 USER project.

Reason, J. (1990). *Human Error*. Cambridge University Press.

Rich, E. (1983). Users are individuals: Individualising user models. *International Journal of Man-Machine Studies*, **18**, 199-214.

Rich, E. (1989). Stereotypes and user modeling. In A. Kobsa and W. Wahlster, editors, *User Models in Dialog Systems*, pages 35-51. Springer-Verlag, Berlin.

Riviere, C. and Thakor, N. (1996). Effects of age and disability on tracking tasks with a computer mouse: Accuracy and linearity. *Journal of Rehabilitation Research and Development*, **33**(1), 6-15.

Roberts, T. L. and Moran, T. P. (1983). The evaluation of text editors: methodology and empirical results. *Communications of the A.C.M.*, **26**, 265-283.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representations by error propagation. In J. McClelland, D. Rumelhart, and the PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.*, chapter 8, pages 318-362. MIT Press, Cambridge, MA.

Schneider-Hufschmidt, M., Kühme, T., and Malinowski, U. (1993). *Adaptive User Interfaces: Principles and Practice*, volume 10 of *Human Factors in Information Technology*. Elsevier Science Publishers B.V., Amsterdam.

Schwartz, P. and Milchus, K. (1992). A simple-to-use software assessment package for testing cursor control ability. In *Resna '92: Proceedings of the 15th Annual Conference*, pages 156-158, Washington D.C. Resna Press.

Sejnowski, T. and Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, **1**, 145-168.

Self, J. (1988). Bypassing the intractable problem of student modelling. In G. Gauthier and C. Frasson, editors, *Proceedings of Intelligent Tutoring Systems '88*, pages 18-24.

Sellen, A. and Nicol, A. (1995). Building user-centered on-line help. In R. Baecker, J. Grudin, W. Buxton, and S. Greenberg, editors, *Readings in Human-Computer Interaction: Toward the Year 2000*, pages 718-723. Morgan Kaufmann Publishers, Inc, San Francisco, CA. Second Edition.

SEMERC (1998). SEMERC catalogue. SEMERC, 1 Broadbent Road, Watersheddings, Oldham, OL1 4LB.

Sentance, S. (1989). Analysing misconceptions in the domain of second language learning. DAI Research paper 586, AI Dept, University of Edinburgh.

Shaw, R., Loomis, A., and Crisman, E. (1995). Input and integration: Enabling technologies for disabled users. In A. D. N. Edwards, editor, *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*, chapter 13, pages 263-278. Cambridge University Press.

Shein, F., McDougall, J., Knysh, B., Sainani, D., Lee, K., Brownlow, N., and Milner, M. (1989a). Guidelines for alternate access system developers. In

Resna '89: Proceedings of the 12th Annual Conference, pages 19-20, Louisiana. Resna Press.

Shein, F., McDougall, J., Knysh, B., Sainani, D., Lee, K., Brownlow, N., and Milner, M. (1989b). A model for alternate access systems. In *Resna '89: Proceedings of the 12th Annual Conference*, pages 17-18, Louisiana. Resna Press.

Sleeman, D. and Brown, J. (1982). *Intelligent Tutoring Systems*. Academic Press Inc, London.

Smutz, P., Serina, E., and Rempel, D. (1994). A system for evaluating the effect of keyboard design on force, posture, comfort and productivity. *Ergonomics*, **37**(10), 1649-1660.

SYSTAT, I. (1992). *SYSTAT: Statistics, Version 5.2 Edition*. SYSTAT, Inc., Evanston, IL.

Thimbleby, H. (1990). *User Interface Design*. ACM Press frontier series. Addison Wesley, New York.

Thimbleby, H. (1995). Treat people like computers? Designing usable systems for special people. In A. D. N. Edwards, editor, *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*, chapter 14, pages 283-295. Cambridge University Press.

Treviranus, J., Shein, F., Hamann, G., Thomas, D., Milner, M., and Parmnes, P. (1990). Modification of direct manipulation pointing devices. In *Resna '90: Proceedings of the 13th Annual Conference*, pages 151-152, Washington D.C. Resna Press.

Trewin, S. (1996). A study of input device manipulation difficulties. In *Proceedings of the Second Annual ACM Conference on Assistive Technologies*, pages 15--22, USA. ACM. ACM ISBN: 0-89791-776-6.

Trewin, S. (1998). InputLogger: General-purpose logging of keyboard and mouse events on an Apple Macintosh. *Behaviour Research Methods, Instruments and Computers*, **30**(2), 327-331.

Trewin, S. and Pain, H. (1997). Dynamic modelling of keyboard skills: Supporting users with motor disabilities. In A. Jameson, C. Paris, and C. Tasso, editors, *User Modeling: Proceedings of the Sixth International Conference*, pages 135-146, New York. Springer Wein. Also available as DAI Research Paper 862.

Trewin, S. and Pain, H. (1998). A model of keyboard configuration requirements. In *Proceedings of the Third ACM Conference on Assistive Technologies*, pages 173-181, USA. ACM.

Vanderheiden, G., Lee, C., and Scadden, L. (1987). Features to increase the accessibility of computers by persons with disabilities: Report from the industry/government task force. In *Resna '87: Proceedings of the 10th Annual Conference*, pages 750-752, Washington D.C. Resna Press.

Vanderheiden, G. C. (1992). Making software more accessible for people with disabilities. University of Wisconsin-Madison. Release 1.2.

Vanderheiden, P. (1985). Writing aids. In J. Webster, A. Cook, W. Tompkins, and G. Vanderheiden, editors, *Electronic Devices for Rehabilitation*, chapter 10, pages 262-282. Chapman and Hall, London. Medical Instrumentation and Clinical Engineering Series.

VanLehn, K. (1988). Student modeling. In M. Polson and J. Richardson, editors, *Foundations of Intelligent Tutoring Systems*, chapter 3, pages 55-78. Lawrence Erlbaum Associates, New Jersey.

Venkatagiri, H. (1993). Efficiency of lexical prediction as a communication acceleration technique. *Augmentative and Alternative Communication*, **9**, 161-167.

Walker, N., Meyer, D., and Smelcer, J. (1993). Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction. *Human Factors*, **35**(3), 431-458.

Watkins, I. (1989). A survey of students with disabilities and their staff supporters using microcomputing equipment in mainstream higher and further education (the COMET awards scheme). In A. T. Vincent, editor, *New Technology, Disability and Special Educational Needs: Some Case Studies*, chapter 8.2, pages 299-308. Open University.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge*. Morgan Kaufmann Publishers, Inc, California.

Zadeh, L. (1994). Fuzzy logic, neural networks and soft computing. *Communications of the A.C.M.*, **37**(3), 77-84.

Zeigler, J. (1996). Interactive techniques. *ACM Computing Surveys*, **28**(1), 185-187.

Glossary

AAC - Augmentative and alternative communication: relating to devices which assist individuals to communicate.

additional key error - A key near to the intended key unintentionally activated by the digit or other part of the body used to activate the desired key. The desired key itself may or may not have been pressed.

alphanumeric key - A key representing a character, number or punctuation mark. Includes space, tab and return. Excludes modifier keys, arrow keys and the delete or backspace key.

assistive technology - Computer or electronic devices and software designed to support the requirements of people with disabilities. Can be used in both daily life or computer access.

Audit - A software utility distributed by Apple, used in the implementation of *InputLogger* and the model of keyboard use.

autoRepeat - The keyboard feature which causes characters to be repeated when a key is held down for long enough. Controlled by *Repeat Keys*.

bounce error - Unintentionally pressing the intended key more than once.

Bounce Keys - A keyboard configuration facility supporting users who are prone to bounce errors. *Bounce Keys* allows the user to introduce a delay after each key press (the debounce time), during which time the same key, if pressed down, will not register, while a different key will register immediately.

configuration - See input device configuration.

configuration facility - A software program which alters the way the keyboard, mouse or some other input device reacts to a given input.

debounce time - The time delay imposed by *Bounce Keys* during which the key last pressed will not register if pressed again.

dropping error - Failure to press two keys simultaneously (e.g. use of the *Shift* key).

gain -See mouse gain.

input device configuration - The process of choosing appropriate settings for the available keyboard and mouse configuration facilities.

InputLogger - A software program for logging keystrokes and mouse events on Macintosh platforms.

key acceptance delay - The length of time a key must be held down for before the key press registers. Can be adjusted with the *Slow Keys* utility.

key repeat delay - The length of time before which a key begins to repeat, once held down.

key repeat rate - The length of time between the generation of repeated characters, once a key that has been held down begins to repeat.

keyboard configuration facility/utility - A software program which alters the way the keyboard reacts to a given input. For example, the key repeat delay can be altered.

keyboard configuration - The process of choosing appropriate settings for the available keyboard configuration facilities.

keyguard - A plastic or metal sheet which fits over the top of the keyboard. It has a hole for each key, and keys are activated by pressing through the appropriate hole.

long key press error - Unwanted repeated characters appearing when an alphanumeric key is pressed for longer than the default key repeat delay.

missing key error - When a movement intended to press a key does not produce a character, either through lack of force or aiming difficulties.

model - A computer program representing some aspect of the current user.

modifier key - A key which generates no character when held down, but changes the effect of other characters. *Shift*, *Control*, *Option* and *Command* are modifier keys.

mouse gain - The distance moved by the cursor for a given distance of mouse movement.

Mouse Keys - A useful keyboard configuration utility for people who can use a keyboard, but not a mouse. It allows users to control the on-screen pointer using the keyboard, including click, double click and drag operations.

Overlap Keys - An experimental configuration facility designed to support slow typists who make many additional key errors by eliminating those errors.

performance error - An error attributable to physical inaccuracy in the manipulation of an input device.

remote error - Character(s) generated while trying to press a key by accidentally pressing a different key with a digit or body part other than the one being used for the intended key press. Other accidental key presses, such as leaning on a part of the keyboard, are also remote errors.

Repeat Keys - A keyboard configuration facility which allows users to control the auto-repeat feature of the keyboard.

Slow Keys - A keyboard configuration facility intended to allow users to eliminate accidental short key presses made, for example, by bumping other keys while moving to a specific key. Usually on a keyboard, a character is registered as soon as a key is pressed, regardless of how long it is pressed for. *Slow Keys* allows the user to introduce a delay, so that only key presses longer than the delay will cause characters to register.

Sticky Keys - A keyboard configuration facility useful for people who find it difficult to press two keys at once. This is generally required when using modifier keys. For example, to produce an asterisk, it is normally necessary to hold down the *Shift* key while pressing '8'. With *Sticky Keys*, modifier keys can be pressed separately, so an asterisk would be typed by pressing and releasing *Shift*, and then pressing '8'.

tick - One sixtieth of a second.

touch typing - A style of keyboard entry which does not require users to look at the keyboard. The hands are held in a defined position over the keyboard, and a specific finger is used to press each key. Both hands and all fingers are used.

transposition error - Two keys are transposed.

Appendix A: Experimental Materials

This appendix contains the task materials and record forms used. Appendix A.1 contains the materials for the initial input error exploration described in Chapter 4. Appendix A.2 contains the task materials and record form for the model evaluation described in Chapter 8.

A.1 Error Exploration Materials

This appendix contains the materials used in the initial data gathering experiment, including the text passage to be copied, the instructions given to the subjects in the mouse and editing tasks, and the passages on which the mouse and editing tasks were performed. Also included is the form on which data concerning each subject was recorded.

The passage used in the typing tasks was originally presented in 18 point type, double spaced. Here it is shown as 12 point with 1.5 line spacing:

There was a British grandfather called Quentin who said that he was 101 years old. ``Are you sure?'' asked his friend Maxine. She was 16.
``Yes! I was born in 1895'' he replied. But Maxine added in her head the sum 1895 + 101 = 1996, and knew that Quentin was actually 100 this year. He is younger than he thinks (but not by much). Perhaps he has forgotten what year it is. I do that sometimes too. Do you?
You may worry about forgetting what the current year is. Zinc in the diet is supposed to help the memory, but who knows!

Passage presented in conjunction with the mouse tasks (originally presented in 18 point type, shown here in 12 point type to retain original line breaks):

1984

******* a ********** April ****** ********** ********** ********** ****** ********** ********* ********* ***** ********** ******** ********* ***** ********** **** ********** ****** ****** ********** ******** ****** *** and *** ****** ***** *********.

The hallway smelt of boiled cabbage and old rag mats. At one end of it a ********** ********** ********** ******* ********** ****** ********** ********** ********* wide: *** ******* ** * *** ** ***** ******* ***** * **** ***** ****** *** *** ******* ** * *** ** ***** **. ****** **** ***. ** *** ** ***** *** *** ***** ****** ********* ****** **** * * **** ** ******** ***** ***** ***** ***** **** **** ** ****** **** daylight *****. ** *** **** ** ***** ******** ****** ****** ****** ****** ******* ******** ******** *** **** ***** ** ***** *** *** ** *** ******** ********* ******* **** *** * so ***** ***** **** *** *** *** * **** ***. BIG BROTHER IS WATCHING YOU, *** ***** **** ***.

********** ********** ********** **** **** ***** ********** ****** ****** ********* ****** * ********** ***** **** ******* *****. It was the police patrol, snooping into people's windows. ******* ******* *** ***** ********** **** ******* * ********** ***** ****** ****** *** *******, however, ******* *******.

Mouse task instructions (originally presented in 18 point type with double spacing between tasks, shown here in 14 point type with single spacing.):

Mouse Tasks

1. Click between the 'i' and 'l' in 'April', on the first line.

2. Drag the cursor to select the word 'WATCHING, in the second paragraph.

3. Double click on the word 'BROTHER' at the end of the second paragraph.

4. Click on the Apple menu at the top left of the screen.

5. Click between the 'i' and 'd' of 'wide', in the second paragraph.

6. Drag the cursor to select the sentence "The hallway smelt of boiled cabbage and old rag mats.", in the second paragraph. NOTE: Include the final full stop.

7. Drag the cursor to select the word 'a' on the first line.

8. Triple click on the first line of the second paragraph.

9. Click on the Application Menu in the top right of the screen.

10. Click the mouse quickly many times without moving it.

11. Drag the cursor to select the phrase: "BIG BROTHER IS WATCHING YOU". NOTE: Do not select the comma at the end of the phrase.

12. Click the mouse where it is, then without moving the cursor, lift the mouse off the table and replace it in a new position. Click again.

13. Drag the cursor to select the whole of the first paragraph

14. Double click on the word 'wide' in the second paragraph.

15. Click between the 'd' and 'a' of 'daylight' in the second paragraph.

16. Drag the cursor to select the word 'and' in the first paragraph.

17. Double click on the word 'so' in the second paragraph.

<End of Mouse Tasks>

Editing task target passage: (The passage was presented on-screen using 18 point type. Here it is shown in 12 point type in order to preserve line break and target positions. The targets are shown here underlined and in bold text. When presented to the subjects, targets were highlighted by using colour instead. The passage is 47 lines long. The screen size used in the experiment allowed 24 lines to be viewed at once.)

1984

It was a bright cold day in **April** and the clocks were striking thirteen. Winston Smith, his chin nuzzled **intpo** his breast in an effort to escape the vile wind, slipped quickly through **teh** glass doors of **Victory Mansions**, though not quickly enough to prevent a swirl of gritty dust from entering along with him.

The hallway smelt of boiled cabbage and old rag mats. At one end of it a coloured poster, too large for indoor display, had been tacked to the wall. It depicted simply **a n** enormous face, more than a metre wide: the face of a **mat** of about forty-five, with a heavy black moustache and ruggedly handsome features. Winston made for the stairs. **It was no use trying the lift**. Even at the best of times it was seldom working, and at present the electric current was cut off during daylight hours. It was part of the economy drive in preparation for Hate Week. The flat was seven flights up, and Winston, who was thirty-nine and had a varicose ulcer above his right ankle, went slowly, resting several times on the way. On each landing, opposite the lift shaft, the poster with the enormous eyes gazes from the **roof**. It was one of those pictures which are so contrived that the eyes follow you about when you move. BIG BROTHER IS WATCHING YOU, the caption beneath it ran.

Inside the flat a fruity voice was reading out a list of figures which had something to do with the production of pig-iron. The voice came from an oblong metal plaque like a dulled mirror which formed part of the surface of the right-hand wall. Winston turned a switch and the voice sank somewhat, though the words were still distinguishable. The instrument (the telescreen, it was called) could be dimmed, but there was no way of shutting it off entirely. He moved over to the window: a smallish, frail figure, the meagreness of his body merely emphasised by the blue overalls which were the uniform of the party. His hair was very fair, his face naturally sanguine, his skin roughened by the course soap, and blunt razor blades and the cold of the winter that had just ended.

Outside, even through the shut window-pane, the world looked cold.

Down in the street little eddies of wind were whirling dust and torn paper into spirals, and though the sun was shining and the sky a harsh blue, there seemd to be no colour in anything, except the posters that were plastered everywhere. The black moustachio'd face gazing down from every commanding corner. There was one on the house-front immediately opposite. BIG BROTHER IS WATCHING YOU, the captionsaid, while the dark eyes looked deep into Winston's own. Down at street level another poster, torn at one corner, flapped fitfully in the wind, alternately covering and uncovering the single word INGOC. In the far distance a helicopter skimmed between the roofs, hovered for an instant like a bluebottle, and darted away again with a curving flight.

Editing task instructions (originally presented in 18 point type with double spacing between tasks, shown here in 12 point type with single spacing):

Editing Tasks

1. Double click on the word "April" in the first line.

2. Press 'Apple' and 'b' at the same time, to make "April" bold.

3. Drag the mouse to select the word "mat" on line 4 of the second paragraph.

4. Type "man", which will replace the selected word.

5. On line 2, change "intpo" to "into".

6. Just below, on line 3, change "teh" to "the".

7. Use the arrows on the scroll bar to move down the document until the top of the second page is exactly at the top of the screen.

8. Click at the end of the text, and type the sentence: "It was the police patrol, snooping into people's windows."

9. Use the box in the scroll bar to move the document back up so that the top of the first page is at the top of the screen.

10. Double click to select 'avobe', half way down the second paragraph, at the right hand side.

11. Type "above", to replace the selected word.

12. Go to the 'Edit' menu, choose 'Find/Change', then 'Find/Change' again.

13. Click on the box in the top left corner of the dialog to close it.

14. Change "roof" into "wall" on the second last line of the second paragraph.

15. Drag the mouse to select "Victory Mansions", in the first paragraph, excluding the following comma.

16. Go to the 'Style' menu, and choose 'Underline'.

17. Use the arrows on the scroll bar to move down the document until the top of the second page is exactly at the top of the screen.

18. Click at the end of the text, and type the sentence: "The patrols did not matter, however."

19. Use the box in the scroll bar to move the document back up so that the top of the first page is at the top of the screen.

20. Drag the mouse to select "It was no use trying the lift.", including the full stop.

21. Go to the 'Style' menu, and choose 'Bold'.

22. Change "a n" into "an", on line 3 of the second paragraph.

23. Go to the 'Outline' menu, choose 'Outline Format', then choose 'Bulleted list'.

24. Go to the 'Outline' menu, choose 'Outline View'.

25. Use the arrows on the scroll bar to move down the document until the top of the second page is exactly at the top of the screen.

26. Click at the end of the text, and type the sentence: "Only the Thought Police mattered."

27. Press 'Apple' and 's' at the same time to save the document.

<End of editing tasks>

Subject record form: (This form was not shown to the subject but filled in by the experimenter.)

Keyboard and Mouse Usability Study

Name _____

Age _____

Computer Experience Beginner Moderate Expert

Word Processing Experience Beginner Moderate Expert

Disability _____

Access Options	Used	Preferred	
key repeat delay	☐	☐	
key repeat rate	☐	☐	
key acceptance delay	☐	☐	
mouse keys	☐	☐	
sticky keys	☐	☐	
keyguard	☐	☐	
other (please specify)	☐	☐	_____

Other Support	Used	Preferred	
wrist rest	☐	☐	
large text size	☐	☐	
magnification software	☐	☐	
audio feedback	☐	☐	
other (please specify)	☐	☐	_____

Type of Mouse	Used	Preferred	
standard	☐	☐	
trackerball	☐	☐	
joystick	☐	☐	
other (please specify)	☐	☐	_____

Mouse Click Speed _____

Mouse Movement _____

Time taken _____

Data Set Number _____

Did you become tired at any point? _____

_____

How easy did you find:

	Impossible	Very Difficult					Easy
Mouse:							
Pointing with the mouse	☐	☐	☐	☐	☐	☐	☐
Clicking the mouse	☐	☐	☐	☐	☐	☐	☐
Dragging with the mouse	☐	☐	☐	☐	☐	☐	☐
Multiple clicking the mouse	☐	☐	☐	☐	☐	☐	☐
Picking up the mouse	☐	☐	☐	☐	☐	☐	☐
Keyboard:							
Pressing two keys at once	☐	☐	☐	☐	☐	☐	☐
Reaching all the keys	☐	☐	☐	☐	☐	☐	☐
Pressing the right key	☐	☐	☐	☐	☐	☐	☐
Pressing only one key	☐	☐	☐	☐	☐	☐	☐
Pressing keys quickly	☐	☐	☐	☐	☐	☐	☐

Was a repeat with preferred configuration done? _____

General Observations:

A.2 Evaluation Materials

The four text passages used in the model evaluation are shown below. These are followed by the form on which data about each participant and the model's recommendations for each participant were recorded.

Passage 1:

It was a dark and stormy night. Mrs. Argyle was talking to her son Jimmy about writing shorthand. She explained that ASAP was a short way of writing 'As Soon As Possible'.

Suddenly there was a knock on the cottage door. Jimmy answered it to a woman wearing a big rosette. "Good evening! Would you like to vote for the Green party?" said the stranger. They invited her in for coffee (she refused their offer of wine) and she told them that 58% of voters were concerned about pollution, and only the Green party could help. To them, 58% seemed rather low - 90% would have been more encouraging.

Passage 2:

The colt from Old Regret had got away. He had joined the wild bush horses. He was worth $1000, so all the best riders had gathered for the chase. There was a $50 reward for the one who caught that colt!

There was Harrison, who made his $8750 fortune when Pardon won the GHIS cup. The old man's hair was very grey, but he would go wherever horse and man could go.

One of them (a young lad) was on a small mountain pony. The old man said "That horse will never stand a long and tiring gallop. What are you doing here?" So they sent the man from Snowy River home and the colt was never caught.

Passage 3:

The military coup in Zambia was hailed as a great success by the new President, General Wilson. He told the nation "The Zambian People's Reform Movement will (by force if necessary) eliminate crime, hunger and the wearing of yellow trousers by the end of the year!"

The families of the 1327 ZPRM soldiers who had died in the battles were awarded compensation of £60 each. How did General Wilson arrive at this figure? It consisted of £15 for funeral expenses and £45 towards the cost of bringing up a new family member.

This was typical of the strange decisions the new government began to make.

Passage 4:

"Is that true?" asked Kevin. His pal had told him of a New Yorker who had jumped from a building, intending to kill himself. Instead he was killed on the way down by a bullet from an apartment window. The charge was homicide.

In the apartment lived Tom and Jan Hill, who had been fighting. He had threatened her with an unloaded (or so he thought) TYJG shotgun.

Their son had loaded it. He was angry because his mum had cut his allowance by 75%, from $100 to $25. He had then become suicidal and jumped from the top of the building. He was killed by his own bullet! The verdict was suicide.

Session Record Form (5 pages):

Session Record Form No:

Name:

Age:

Keyboard experience:

Length of time:
Machine(s) used:
Frequency of use:

Disability:

Typing style:

Other keyboard support:

Name of tutor/independent assessor:

Order of passages:

Fatigue/pain:

General observations:

Sticky Keys

Activated in: P2 P3 P4 None
Passage used: G Z C N

Initial Opinion:

Awareness: Unknown / Not sure / Known
Availability: Unavailable / Unknown / Available
Frequency: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Independent expert's Opinion:

Awareness: Unknown / Not sure / Known
Availability: Unavailable / Unknown / Available
Frequency: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Observation: No Maybe Yes

Modeller (P1): No Maybe Yes
Modeller (P2): No Maybe Yes
Modeller (P3): No Maybe Yes
Modeller (P4): No Maybe Yes

Final Opinion:

Uptake: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Notes:

Repeat Keys

Activated in:	P2	P3	P4	None	
Passage used:	G	Z	C	N	
Delay Setting used:	Off	40	24	16	12

Initial Opinion:

Awareness: Unknown / Not sure / Known

Availability: Unavailable / Unknown / Available

Usefulness: Not / Somewhat / Useful / Very / Essential

Preferred Setting: Off / 40 / 24 / 16 / 12 / Default

Independent expert's Opinion:

Awareness: Unknown / Not sure / Known

Availability: Unavailable / Unknown / Available

Usefulness: Not / Somewhat / Useful / Very / Essential

Recommended Setting: Off / 40 / 24 / 16 / 12 / Default

Observation: Off / 40 / 24 / 16 / 12 / Default

Modeller (P1): Off / 40 / 24 / 16 / 12
Modeller (P2): Off / 40 / 24 / 16 / 12
Modeller (P3): Off / 40 / 24 / 16 / 12
Modeller (P4): Off / 40 / 24 / 16 / 12

Final Opinion:

Usefulness: Not / Somewhat / Useful / Very / Essential

Preferred Setting: Off / 40 / 24 / 16 / 12 / Default

Notes:

Overlap Keys

Activated in: P2 P3 P4 None
Passage used: G Z C N

Initial Opinion:

Frequency: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Independent expert's Opinion:

Frequency: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Observation: No Maybe Yes

AD Errors(P1):
AD Errors(P2):
AD Errors(P3):
AD Errors(P4):

Overlaps (P1):
Overlaps (P2):
Overlaps (P3):
Overlaps (P4):

Final Opinion:

Uptake: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Notes:

Bounce Keys

Initial Opinion:

Awareness: Unknown / Not sure / Known
Availability: Unavailable / Unknown / Available
Frequency: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Independent expert's Opinion:

Awareness: Unknown / Not sure / Known
Availability: Unavailable / Unknown / Available
Frequency: Never /Rarely /Sometimes /Often /Always
Usefulness: Not / Somewhat / Useful / Very / Essential

Observation: No Maybe Yes

Evidence (P1):
Evidence (P2):
Evidence (P3):
Evidence (P4):

Notes:

Appendix B: Log File Formats

Automatic logging of input events, combined with manual annotation of the log files, allowed much of the analysis of the keyboard and mouse usage data described in Chapter 4 to be automated. A similar procedure was used in analysis of the data gathered during evaluation of the model. This appendix specifies and gives examples of the four kinds of log file that were used: raw data from *InputLogger*, annotated data from *InputLogger*, raw data from the model, and annotated data from the model.

B.1 Raw data from InputLogger

The events recorded in the log file are exactly those recognised by the Macintosh operating system: `KeyDown`, `KeyUp`, `KeyRepeat`, `MouseDown`, and `MouseUp`, with the addition of one new event type: `MouseMove`. A `MouseMove` event is recorded whenever the mouse changes direction by more than 10 degrees. This provides an accurate yet compact record of the mouse path taken between mouse clicks, including small-scale shake and large scale positioning movements. The default value of 10 degrees can be altered on the *InputLogger* control panel.

Due to technical restrictions imposed by the Macintosh operating system, and the application-independent nature of *InputLogger*, `MouseMove` events are not recorded while the mouse button is held down. With the exception of *Caps Lock*, presses of modifier keys are recorded as `KeyDown`/`KeyUp` pairs, in the same way as other key presses. Because the *Caps Lock* key is a latching key, the logging mechanism cannot record the length of time for which the key is actually held down each time it is pressed. Instead, a single `KeyDown` or `KeyUp` event is recorded for each press of *Caps Lock*. `KeyRepeat` events are not generated for modifier keys.

An example log file is shown in Figure B.1. Each log file entry is of the form:

```
<nLost> <time> <eventType> <eventData>
```

- `<nLost>` – is a check for internal logging failures, which should remain zero. No logging failures occurred during data gathering.

- `<time>` - the time at which the event occurred. Time is measured as the number of *ticks* since system start-up. A tick is a sixtieth of a second.
- `<eventType>` - specifies what event has occurred. One of: `KeyDown (kd)`, `KeyUp (ku)`, `KeyRepeat (kh)`, `MouseDown (md)`, `MouseUp (mu)`, `MouseMove (mm)`.
- `<eventData>` - For keyboard events, the event data recorded is the hexadecimal ASCII code for the relevant key. Unused ASCII codes are employed to specify keys such as *Shift* which have no ASCII code of their own. For mouse events, event data is the mouse position on the screen, recorded as separate x and y co-ordinates.

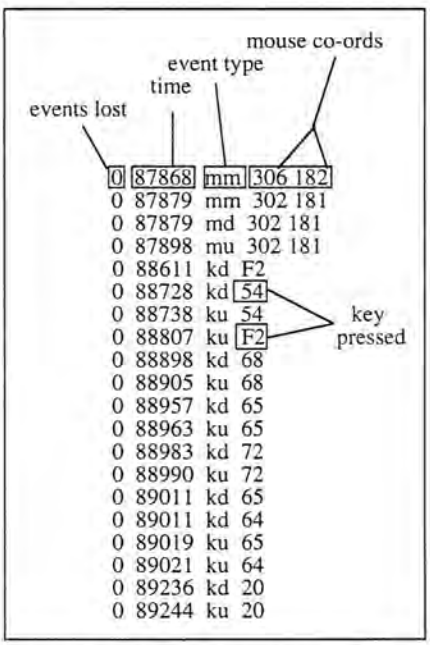


Figure B.1: A raw InputLogger log file

B.2 Annotated data from InputLogger

The raw log files produced by *InputLogger* were processed into a more readable form and then annotated by hand, using task knowledge, observations and video evidence to identify different types of error, targets of mouse movement and other events such as correction of errors.

In the annotated log files, each line is of the form:

```
<time> <eventType> <eventClass> <eventData>
```

- `<time>` - the time at which the event occurred, unchanged from the raw log file.

- `<eventType>` - specifies what event has occurred. One of: `keyDown` (kd), `keyUp` (ku), `keyMissing` (km), `mouseDown` (md), `mouseUp` (mu), `mouseMove` (mm) or `control` (cc). `KeyRepeat` events are removed, as the key repeat delay in force was known, so they do not add any information. `keyMissing` and `control` events have been added. A `keyMissing` event occurs when the task demanded a given key press which did not occur. This may have been because the user made no attempt to press the key or keys, or because the user's attempt failed.

- `<eventClass>` - within each event type, a number of different event classes are defined. For example, a key down event could be correct, or one of a number of different errors. A control event could contain information about error correction or cursor positioning. The event classes associated with each event type are summarised in Table B.1 and each is further clarified below.

- `<eventData>` - Up to three items of data associated with the event appear at the end of the line. Data recorded includes the character corresponding to a `keyDown` or `keyUp` event, the co-ordinates of a mouse event, and information about associations between intended and unwanted characters. The event data associated with each event class is summarised in Table B.1 and detailed below.

Table B.1: Summary of annotated InputLogger log file entries

Event Type	Event Class	Event Data	Interpretation
kd	co	keyHit	A correct key press
	ca	keyHit numOtherHits	Correct, but not the only key activated
	dw	keyHit	A deliberate wrong key press
	da	keyHit numOtherHits	Wrong key, not the only key activated
	ad	keyHit keyIntended time	An additional key error
	re	keyHit keyIntended time	A remote key error
	tr	keyHit intendedPosn	A transposed key
	bo	keyHit	A bounce error
ku	co	keyRaised	A key raised deliberately
	dr	keyRaised	A key raised unintentionally
	bo	keyRaised	A bounce on *Caps Lock*
km	om		Text in the task was omitted here
	mi	keyMissed	A failed attempt to press a key
	ma	keyMissed numOtherHits	Failed key press, other key(s) activated
md	sc	coords target clickType	Start of a click
	ic	coords	Within a multiple click
	sd	coords target	Start of a drag
	cd	coords target	Continuing a drag
	ac	coords	An accidental click
mu	ec	coords	End of a click
	ic	coords	Within a multiple click
	ed	coords target upClass	End of a drag
	ei	coords	'Ghost' click end
mm	mm	coords	A mouse movement
	ic	coords	Mouse movement within a click
cc	ps		Start correcting a performance error
	pe		Stop correcting a performance error
	es		Start correcting other errors
	ee		Stop correcting other errors
	ws	coords target	The text window was shifted

B.2.1 KeyDown Events

There are eight kinds of `keyDown` event, all of which have a string associated with them, representing the key that has been pressed down. For the majority of keys, the string is a single character. For modifier keys, arrow keys, the backspace key, clear, tab, return and space, a longer string is used, for example: "<tab>". A `keyDown` event is classified as:

- correct and free of performance errors; correct but with associated additional key or remote errors;

- incorrect with respect to the task but free of performance errors;
- incorrect and having associated additional key or remote errors;
- an additional key error;
- a remote error;
- a transposition error;
- or a bounce error.

Keystrokes with associated errors carry event data giving the number of performance errors associated with the keystroke. This number is usually one, but it is possible for many additional key or remote errors to be associated with a single keystroke. Additional and remote errors also carry additional data, giving the character the user intended to produce at the time of pressing the key (for remote errors this could be a null character) and the direction in time (forwards or backwards) of the `keyDown` event of the intended key press. Finally, transposition errors also carry data indicating whether they are transposed with a key ahead or behind in the input stream.

B.2.2 KeyUp events

There are three classes of `keyUp` event, all of which have a string representing the key pressed associated with them. The possible `keyUp` events are:

- a key has been deliberately raised;
- a key has been raised without the user intending it - a dropping error;
- or a bounce error. This option is only possible for the *Caps Lock* key. Due to the latching action of this key, a bounce error which activates and deactivates the latch, can only be recorded on the `keyUp` event of the bounced key press. In theory it would also be possible for `keyUp` events on *Caps Lock* to represent other errors such as additional or remote errors. In practice this was not observed in the data.

B.2.3 KeyMissing events

A new type of event not present in the raw data logs produced by *InputLogger* is the `keyMissing` event. There are three classes of `keyMissing` event:

- an omission, in which the participant made no attempt to type one or more characters of the text presented in the typing tasks;
- a missing key, where the participant made an attempt to press a key but the key did not register, either because it was not pressed hard enough, or because the aim of the key press was inaccurate;
- a missing key with associated performance errors, where the participant attempted to press a key, failed to activate the intended key, but succeeded in activating one or more other keys, causing additional key or remote errors.

Missing (as opposed to omitted) keys have event data indicating the intended key. Where associated performance errors were made, the event data also indicates the number of associated unwanted key presses.

B.2.4 MouseDown events

The five categories of `mouseDown` event represent:

- the start of a click;
- the start of the second or further clicks in a double, triple or multiple click;
- the start of a drag operation;
- the continuation of a drag operation where the participant accidentally released the pressure on the mouse button, and then reapplied it;
- and the start of an accidental click, where the participant did not intend to press the mouse button.

All of these events record the mouse co-ordinates as event data. In addition, the start of a click or a drag has a target number to aid automatic evaluation of pointing accuracy, drag continuations have a null target, and the start of a click has an integer

representing the type of click: 1 for a single, 2 for a double, 3 for a triple and 4 for a multiple click.

B.2.5 MouseUp events

There are four classes of `mouseUp` event. These represent:

- the end of a click;
- a `mouseUp` within a double, triple or multiple click;
- the end of a drag;
- and a 'ghost' `mouseUp` event. This is inserted when the participant has clicked outside the text window, activating the Finder, and the clicked back in again - the Macintosh operating system does not necessarily return both of the desired `mouseUp` events when this happens, and so a ghost event is inserted to ease processing of the logs.

All of the `mouseUp` events have a pair of mouse co-ordinates associated with them. In addition, when a drag operation is completed, the event data indicates the target of the end of the drag, and the class of the end of the drag. This class gives an indication of the reason why the drag was ended, and is an integer representing one of the states: deliberate release of the mouse button to complete drag; deliberate release of the mouse button to abandon drag; or accidental release of the mouse button (a dropping error).

B.2.6 MouseMove events

Mouse movements are classed as being:

- normal movements of the pointer;
- or movements within a click.

They have only the mouse co-ordinates recorded as event data.

B.2.7 Control events

Control events are annotations indicating events relevant to the processing of the log files, rather than events caused by the participant. There are five such events:

- the start of a period spent correcting a performance error;
- the end of a period spent correcting a performance error;
- the start of a period spent correcting a non-performance error;
- the end of a period spent correcting a non-performance error;
- a 'window shift' event. These are relevant only in the mouse and editing tasks. They indicate that the text window has been moved, and therefore the targets at which the user is pointing are in a new screen position. Data associated with these events gives a co-ordinate pair and a target number, and indicate the new position of the target. A target number of zero indicates that the co-ordinates represent the relative movement of the window from its previous position, and this is the form most often used in the annotation. The target window was never deliberately moved from its original position, but several participants moved it unintentionally, generally through accidental clicks.

In mouse pointing, clicking and dragging tasks, errors did not require correction, simply a repeat attempt. The first four of these annotations therefore appear only in the text copying tasks, and the text based parts of the editing task. They have no event data associated with them. The time given for the start of a correction period is the time of the first input event the user makes as part of the correction. The time of the end of the period is the time of the last correction event. The software processing the log files calculates the total time between these two events. The calculation of time spent making corrections therefore excludes time spent checking for errors and deciding what corrective actions to take. It also excludes the time spent returning the insertion point to the desired position after making the correction.

B.2.8 Example log file

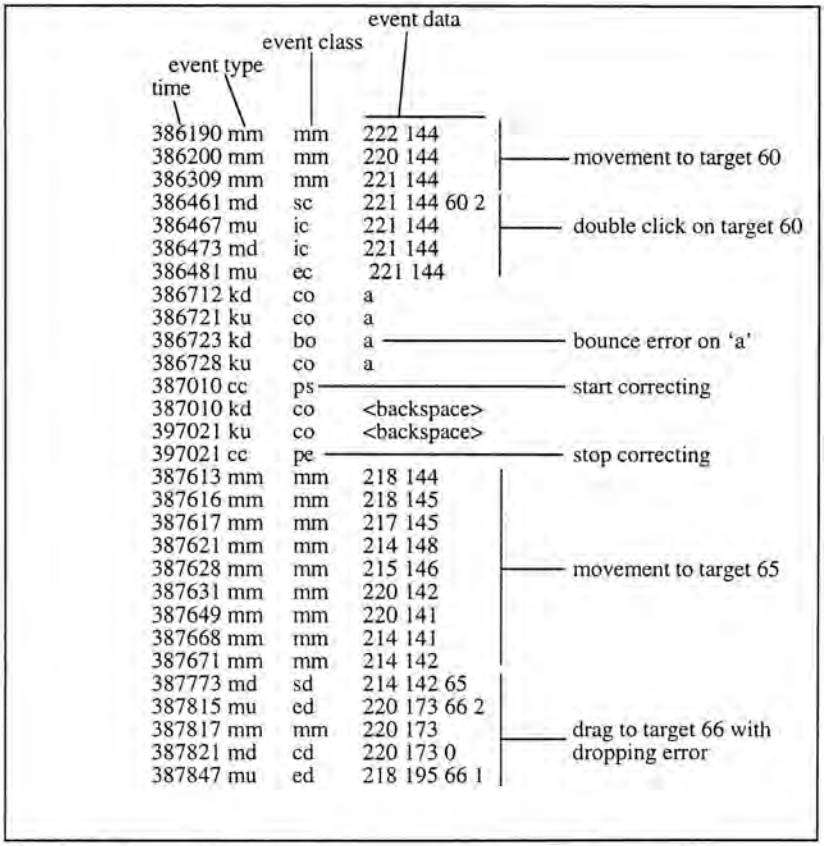


Figure B.2: Example of an annotated InputLogger file

Figure B.2 shows an example log file fragment from the editing task, in which the participant double clicks on target '60', types the letter 'a', making a bounce error, deletes the extra character, then performs a drag from target '65' to target '66', making a dropping error.

B.3 Raw data from model evaluation

Evaluation of the model produced input log files similar to those written by *InputLogger*. `KeyRepeat` and all mouse events were not recorded, and a number of new event types representing changes in the model were added. As before, the basic form of each line in the log file is:

```
<nLost> <time> <eventType> <eventData>
```

- `<nLost>` - is a check for internal logging failures, which should remain zero. No logging failures occurred during data gathering.
- `<time>` - the time at which the event occurred, measured as in *InputLogger*.
- `<eventType>` - specifies what event has occurred. One of: `KeyDown (kd/cd)`, `KeyUp (ku/cu)`, `ModelStart (ms)`, `ModelEnd (me)`, `RepeatKeys (rk)`, `StickyKeys (sk)`, `BounceKeys (bk)`, or `OverlapKeys (ok)`.
- `<eventData>` - For keyboard events, the event data recorded is the hexadecimal ASCII code for the relevant key. As before, unused ASCII codes are employed to specify keys such as *Shift* which have no ASCII code of their own. Model start and end events indicate when the model is initialised and stopped, and carry no event data. For `RepeatKeys` events, event data is the recommended repeat delay, the average keystroke length, and the number of keystrokes on which the recommendation is based. For `StickyKeys`, event data is the current setting chosen for *Sticky Keys* (on, maybe or off), and the total evidence value. For `BounceKeys` events, event data is the recommended bounce delay (0 indicates no delay) and the current bounce keys evidence value. Finally, for `OverlapKeys`, the data recorded is the model's count of the number of additional key errors and the number of deliberate overlapping key presses observed since modelling began.

Figure B.3 shows an example log file. Keyboard events are recorded whenever they occur, the source recording the event (control panel of system extension) is indicated by the use of 'k' and 'c': 'k' for system extension, 'c' for the control panel. All alphanumeric keystrokes are recorded by the system extension. Modifier key events

can be recorded by either the system extension or the control panel, usually both. Where modifier key events are duplicated, the earlier event (typically the one recorded by the control panel) is the most accurate.

Model events are recorded when logging is started or stopped, when modelling is started or stopped, and whenever the event data values change.

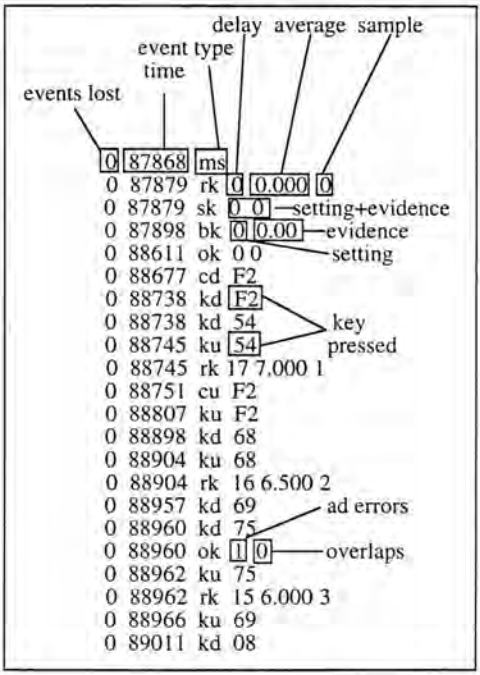


Figure B.3: Example log file produced by the model

B.4 Annotated data from model evaluation

Like the raw *InputLogger* log files, the log files produced by the model were processed into a more readable form and then annotated by hand, using task

knowledge, observations and video evidence to identify different types of error, and other events such as correction of errors.

As before, each line in the annotated log files is of the form:

```
<time> <eventType> <eventClass> <eventData>
```

- `<time>` - the time at which the event occurred, unchanged from the raw log file.
- `<eventType>` - specifies what event has occurred. One of: keyDown (kd), keyUp (ku), keyMissing (km), control (cc), or model (nn). Keyboard and model events are directly translated from the raw log file, while control events are added during the annotation process. Some keyboard events are also modified during annotation, to add information about errors.
- `<eventClass>` - the event classes within each type are the same as those described for annotated InputLogger files, except that mouse events have been removed, control events related to mouse usage have been removed, control events related to the use of *Sticky Keys* and *Overlap Keys* have been added, and model events have been added. The event classes associated with each event type are summarised in Table B.2 and each is further clarified below.
- `<eventData>` - Up to three items of data associated with the event appear at the end of the line. Data recorded includes the character corresponding to a keyDown or keyUp event, the current settings of the model, and information about associations between intended and unwanted characters. The event data associated with each event class is summarised in Table B.2 and detailed below.

Figure B.4 shows the annotated version of the log file fragment shown in the previous section.

KeyDown, KeyUp and KeyMissing events are identical to those described for annotated *InputLogger* files, with the exception that remote key errors are no longer associated with a specific keystroke. This change was made because during annotation of the *InputLogger* data it was found that the majority of remote errors were not associated with any deliberate key press, remote errors were not of interest to the model, and it simplified the annotation process.

Table B.2: Summary of event classes and associated event data for annotated log files produced by the model

Event Type	Event Class	Event Data	Interpretation
kd	co	keyHit	A correct key press
	ca	keyHit numOtherHits	Correct, but not the only key activated
	dw	keyHit	A deliberate wrong key press
	da	keyHit numOtherHits	Wrong key, not the only key activated
	ad	keyHit keyIntended time	An additional key error
	re	keyHit	A remote key error
	tr	keyHit intendedPosn	A transposed key
	bo	keyHit	A bounce error
ku	co	keyRaised	A key raised deliberately
	dr	keyRaised	A key raised unintentionally
	bo	keyRaised	A bounce on *Caps Lock*
km	om		Text in the task was omitted here
	mi	keyMissed	A failed attempt to press a key
	ma	keyMissed numOtherHits	Failed key press, other key(s) activated
cc	ps	errorType	Start correcting a performance error
	pe		Stop correcting a performance error
	es		Start correcting other errors
	ee		Stop correcting other errors
	ss		Start correcting *Sticky Keys* errors
	se		Stop correcting *Sticky Keys* errors
	os		Start correcting *Overlap Keys* errors
	oe		Stop correcting *Overlap Keys* errors
nn	ms		(Re)start modelling
	rk	setting pressLen sample	Update *Repeat Keys* recommendation
	sk	setting evidence	Update *Sticky Keys* recommendation
	ok	nAdErrors nOverlaps	Update additional key error or deliberate overlap counts
	bk	setting evidence	Update *Bounce Keys* recommendation

The set of control events has been extended to include markers for time spent correcting errors caused by the use of *Sticky Keys* (for example, when a user accidentally activated the locking mechanism) and *Overlap Keys* (time spent reinserting characters which had been wrongly deleted).

Model events are carried through from the raw log data, and the event data associated with them is as described in the previous section.

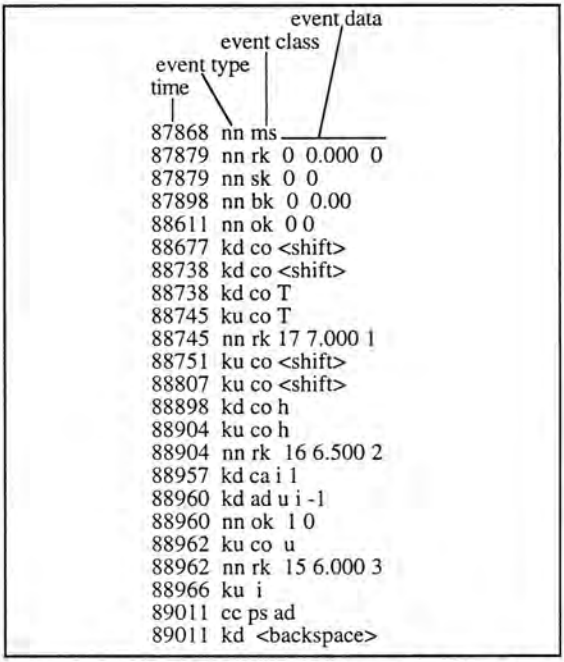


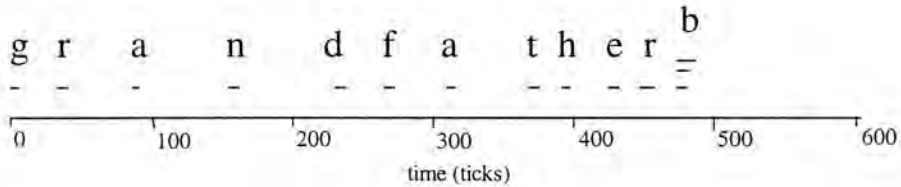
Figure B.4: Annotated version of a model log file

Appendix C: Further Example Input Patterns

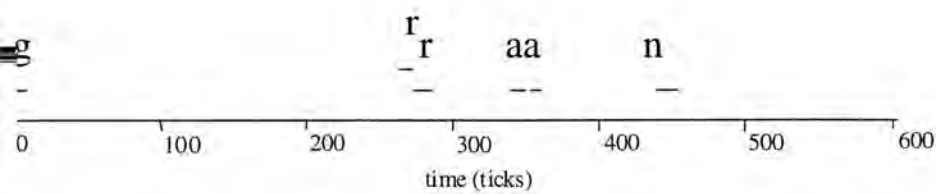
C.1 Keyboard Input Examples

The following examples show the typing patterns recorded for some of the participants in the study described in Chapter 4. Each diagram shows a time line, measured in ticks, representing approximately ten seconds of typing, starting from a point near the beginning of the passage. The text to be typed is: "grandfather called Quentin who said that". The keystrokes are shown by dashes, with the appropriate characters above. The *Shift* key is represented by '*', and the space is represented by '_'.

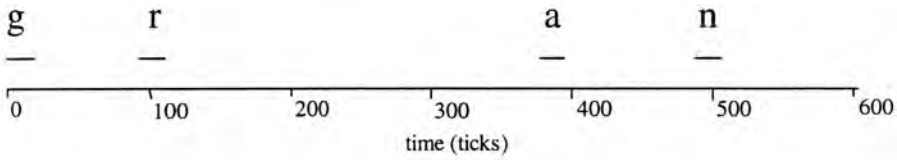
Participant 1, second typing task



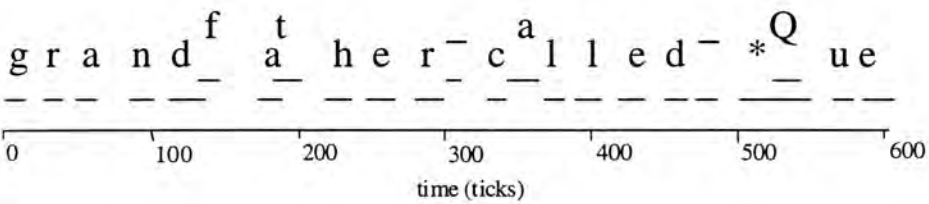
Participant 6, first typing task



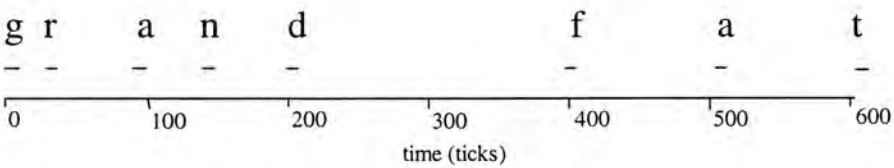
Participant 13, first typing task



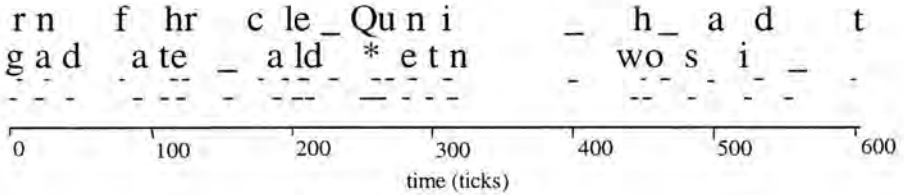
Participant 19, second typing task



Participant C1, first typing task



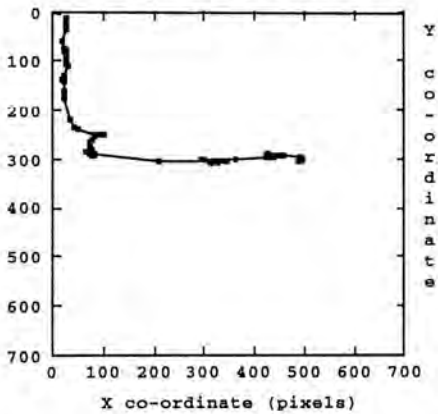
Participant C2, second typing task



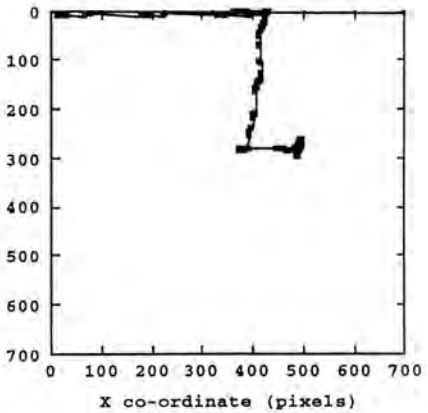
C.2 Mouse Paths when Pointing

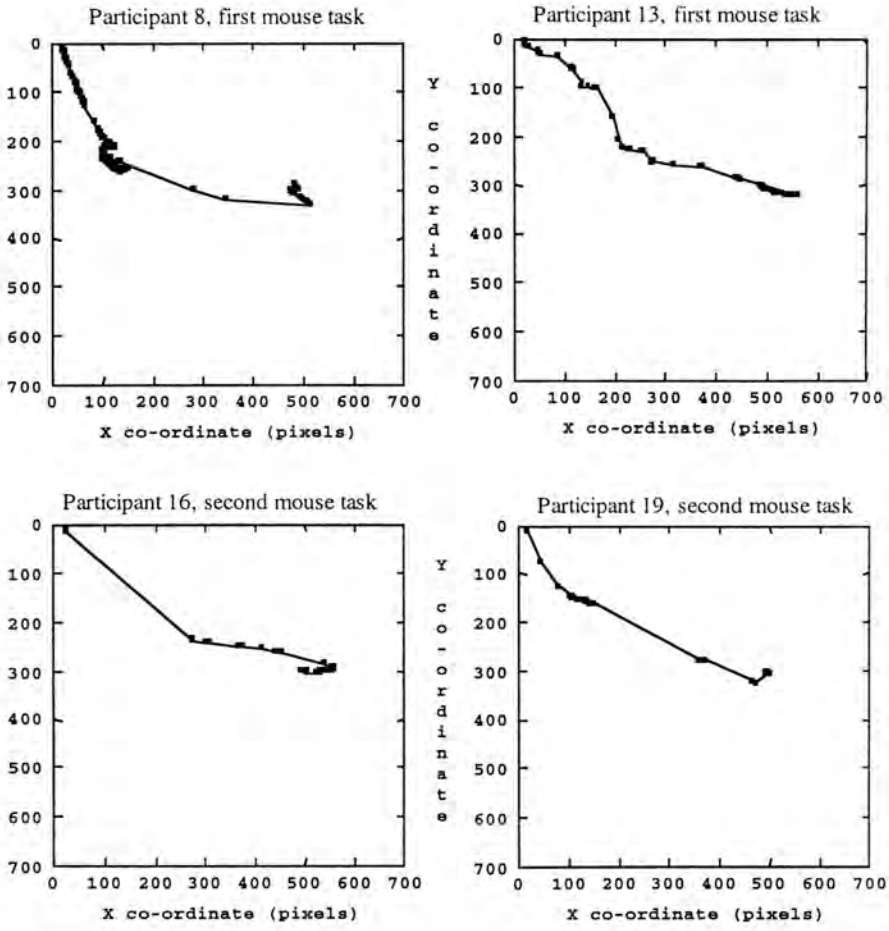
The following graphs show the path taken by the pointer when moving from the top left of the screen to a target near the centre, for a number of different participants.

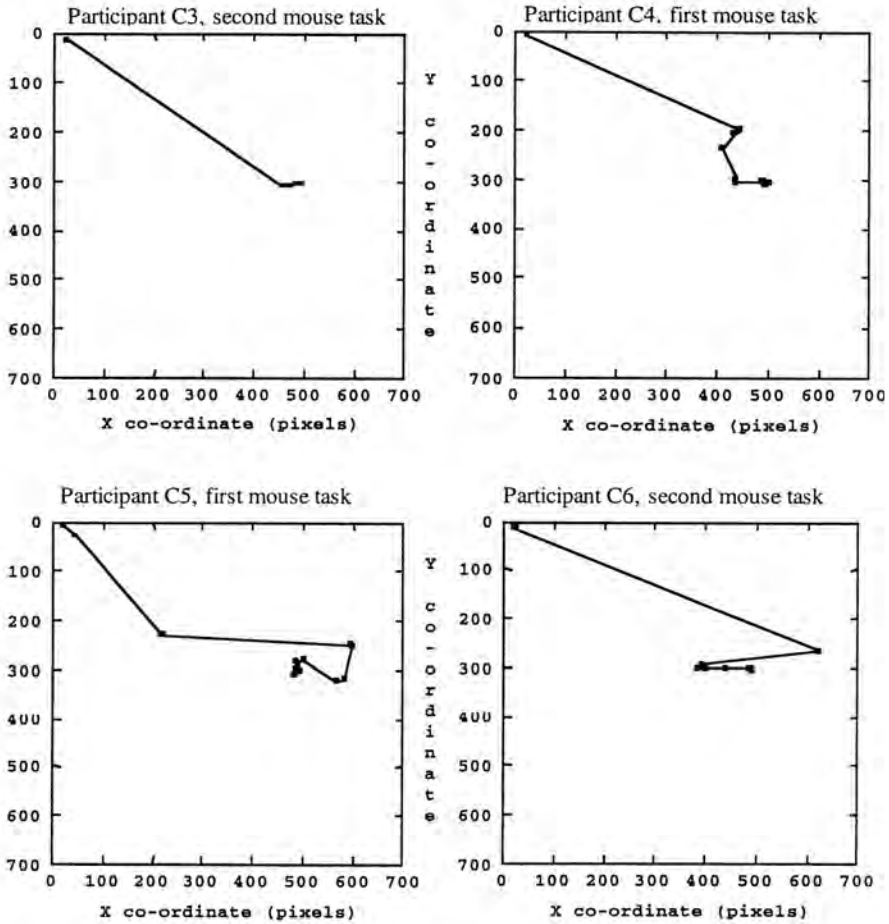
Participant 4, second mouse task



Participant 7, first mouse task

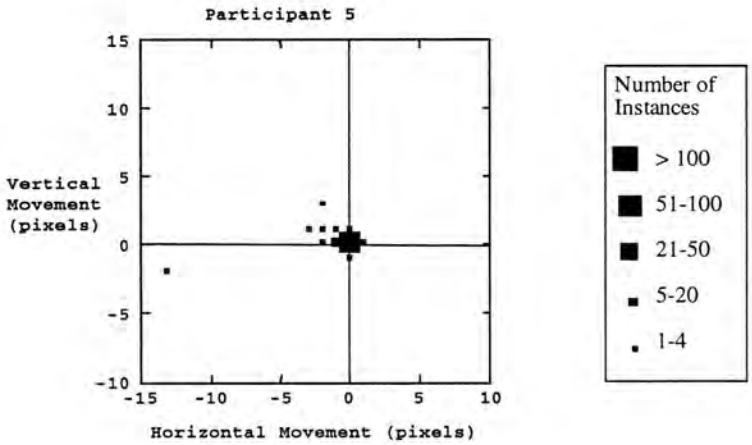


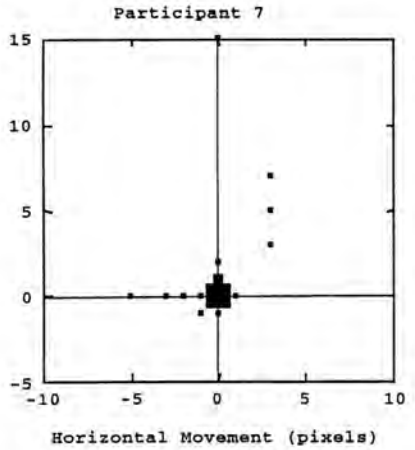

Participant 8, first mouse task
Participant 13, first mouse task
Participant 16, second mouse task
Participant 19, second mouse task
Y co-ordinate
X co-ordinate (pixels)
0
100
200
300
400
500
600
700

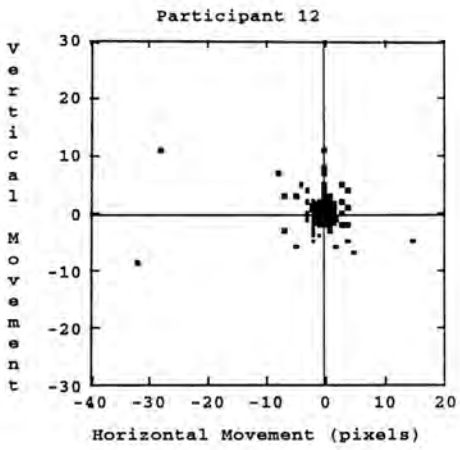

Participant C3, second mouse task
Participant C4, first mouse task
Participant C5, first mouse task
Participant C6, second mouse task
Y co-ordinate
X co-ordinate (pixels)
0
100
200
300
400
500
600
700

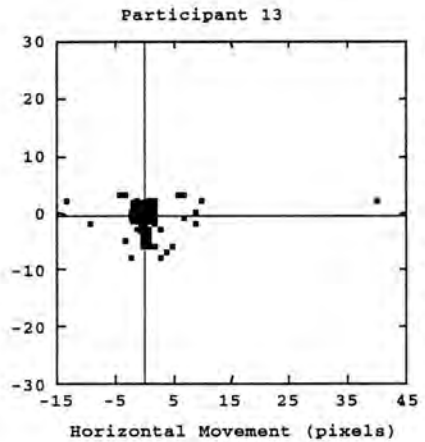
C.3 Click Movement Patterns

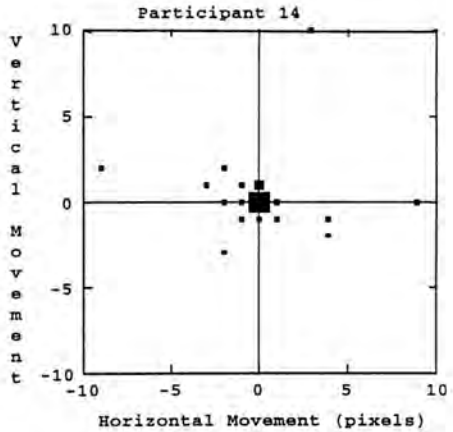
The following graphs provide some further illustration of patterns of movement within clicks for the participants with motor disabilities. As mentioned previously, some exhibit directional patterns, while others do not.

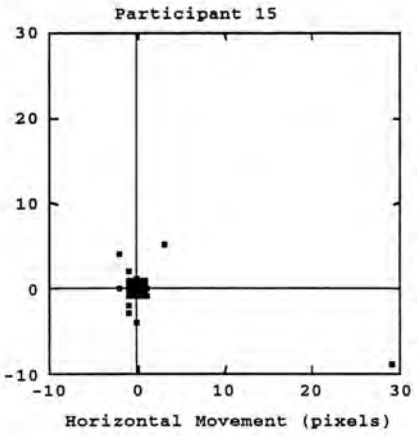


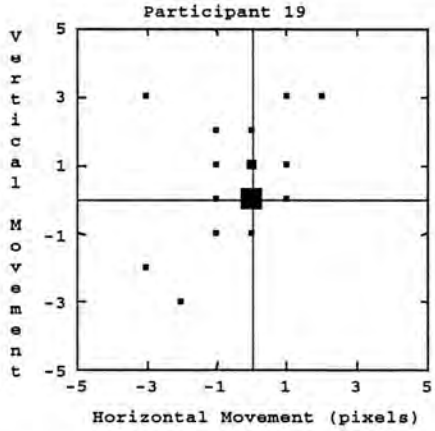

Participant 7
15
10
5
0
-5
-10
-5
0
5
10
Horizontal Movement (pixels)

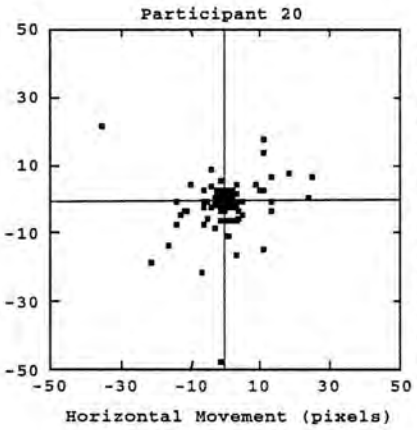

Participant 12
Vertical Movement
30
20
10
0
-10
-20
-30
-40
-30
-20
-10
0
10
20
Horizontal Movement (pixels)

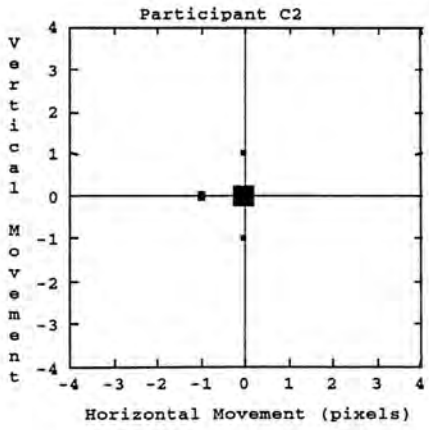

Participant 13
30
20
10
0
-10
-20
-30
-15
-5
5
15
25
35
45
Horizontal Movement (pixels)


Participant 14
Vertical Movement
10
5
0
-5
-10
-10
-5
0
5
10
Horizontal Movement (pixels)


Participant 15
30
20
10
0
-10
-10
0
10
20
30
Horizontal Movement (pixels)


Participant 19
Vertical Movement
5
3
1
-1
-3
-5
-5
-3
-1
1
3
5
Horizontal Movement (pixels)


Participant 20
50
30
10
-10
-30
-50
-50
-30
-10
10
30
50
Horizontal Movement (pixels)


Participant C2
Vertical Movement
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4
Horizontal Movement (pixels)

Appendix D: Published Papers

Some of the work described in this thesis has already been published elsewhere. This appendix reproduces the four existing published papers, with permissions from the publishers.

The four papers are:

Trewin, S. (1996) A study of input device manipulation difficulties. *Proceedings of the Second Annual ACM Conference on Assistive Technologies*, 15-22, USA, ACM. (Copyright 1996 The Association for Computing Machinery)

Trewin, S. and Pain, H. (1997) Dynamic Modelling of Keyboard Skills: Supporting Users with Motor Disabilities. In *User Modeling: Proceedings of the 6th International Conference*, A. Jameson, C. Paris and C. Tasso, Eds. Springer Wein, New York, pp 135-146. (Copyright 1997 Springer Wein New York)

Trewin, S. and Pain, H. (1998) A model of keyboard configuration requirements. *Proceedings of the Third ACM Conference on Assistive Technologies*, pages 173-181, USA, ACM. (Copyright 1998 The Association for Computing Machinery)

Trewin, S. (1998) InputLogger: General-Purpose Logging of Keyboard and Mouse Events on an Apple Macintosh. *Behavior Research Methods, Instruments & Computers*, **30**(2), 327-331. (Copyright 1998 The Psychonomic Society)

The papers are reproduced in the order shown above.

A Study of Input Device Manipulation Difficulties

Shari Trewin
University of Edinburgh
Department of Artificial Intelligence
80 South Bridge
Edinburgh
EH1 1HN
Scotland
shari@aisb.ed.ac.uk

ABSTRACT

'eople with a motor disability affecting their use of the key-oard and/or mouse often tend to make unintentional input rrors. Little or no quantified data exists on physical errors in ıe use of standard computer input devices, particularly with :spect to motor disabilities.

uch information, if available, could be used to develop tech-iques for automatic recognition of specific difficulties. Once cognised, many can be reduced or eliminated by appropri-e system and application configuration.

his paper describes the pilot study for an experiment in-nded to gather detailed information about input errors made ith keyboards and mice. This work is a step towards provi-on of dynamic, automatic support for the configuration of stems and applications to suit individual users.

ıme initial results from the pilot study are presented, includ-g an assessment of the experiment design and a summary some interesting characteristics of the data gathered so far.

EYWORDS: keyboard, mouse, errors, physical disability, put devices, input logging.

TRODUCTION

is widely recognised that people with physical disabili-s can have difficulty in accurately manipulating the stan-rd computer input devices: the QWERTY keyboard and : mouse. However, despite the fact that a vast num-r of access facilities and alternative input devices have :n developed to get around these difficulties [Brown, 1992] ALL Centre, 1994], very little quantified data on their pre-e nature exists.

ysical errors in the manipulation of input devices are re-red to here as *performance errors*, to distinguish them from gnitive and other errors. Examples of performance errors are missing keys, striking adjacent keys in addition to the one aimed for, pressing keys for too long producing repeated letters, moving the mouse while double clicking, and dropping the mouse button while dragging.

It is conjectured that particular performance errors caused by physical disabilities will exhibit patterns which can be recognised. For example, where a key adjacent to the intended key is additionally struck, the timings of the intended and accidental key presses may distinguish them from those of deliberate, separate key presses. If this conjecture is true for several different performance errors, then the information gleaned can be used to provide support for users experiencing difficulties of this type.

This support could, for example, include the configuration of input devices to accommodate performance errors in the input stream. There are a number of configuration options available for keyboards and mice which can reduce or eliminate some varieties of performance error. If the need for one of these options could be dynamically recognised, then the option could be activated automatically, improving the usability of the computer system as a whole. Machine support for configuration is an attractive proposition, since an individual's needs may change dramatically over time, and human support is often in short supply.

Before an automatic configuration tool, or similar support can be developed, a good understanding of the identifying characteristics of performance errors in an input stream is required. An investigative study is proposed, the goals of which are: to examine performance errors from a number of different users with disabilities, producing data on the nature and frequencies of performance errors occurring; and to test the hypothesis that these performance errors are often characterised by recognisable patterns in the input stream.

This paper describes the pilot test for the study, and summarises some preliminary results obtained. The next section explains the lack of existing data of this kind, and the consequent need for this study. The study itself is described, followed by details of the technique used to log the input events generated by the subjects. Initial results obtained from the data gathered, and some preliminary conclusions from these observations are presented. The paper concludes with a summary of the implications for the design of the full study, and

TS '96, Vancouver, British Columbia, Canada
6 ACM 0-89791-776-6/96/04..$3.50

for the prospects of applying these results to improve the usability of standard input devices.

EXISTING DATA

Even for non-disabled people, data on physical input errors is difficult to find. HCI research into human errors has generally focussed on *cognitive* errors and their causes, ignoring physical errors of the kind produced by erratic motor control. This despite evidence that 'keyboarding errors' are an important and significant source of errors, particularly in large databases [Peterson, 1980].

Finding data produced by people with disabilities is even more difficult. Evaluation of keyboards, mice and applications is almost invariably carried out either with expert users, or with users who are typical of the intended user population, or of sub-classes of that population [Karat, 1988] [Vassiliou, 1984] [Monk, 1987]. Disabled users cannot be viewed as a homogeneous class of users in this way. Proportionally, the expected number of physically disabled users of a general purpose system is small, and because their consideration in the evaluation process is difficult to justify economically, they are rarely considered.

One might expect that appropriate data is available from the evaluation of the computing needs of people with physical disabilities. However, when people are assessed in order to establish an appropriate input device and software, their keyboard and mouse skills are assessed by observation, and usage data is not recorded [Broadbent & Curran, 1992] [Lee & Thomas, 1990]. In fact, the author is not aware of any input data recorded for physically disabled users of keyboards and mice.

Real input data is essential in order to be able to establish what errors do occur, and their relative frequencies. It will also allow examination of associations between specific errors and features of the stream of input events. The goal of the proposed study is to gather some accurate data from which this information can be extracted.

STUDY DESCRIPTION

Before launching into a major study of performance errors, it is important to ensure that the proposed experimental method and the data collection tools are adequate. To this end, a small pilot study was carried out. This test allowed the experiment design and data collection software to be assessed and refined prior to the major study. This section describes the methodology proposed for the study, and tested in the pilot study.

Subjects

The subjects chosen for the full study will form a set of case studies, rather than a representative cross-section of people with motor disabilities. The study will not focus on any particular disability, but aims to cover a broad range of people, experiencing a variety of keyboard and mouse difficulties. It is anticipated that there will be overlap between the sets of difficulties caused by different disabilities.

The intention of the study is to capture the normal performance of each user, and so it is essential to minimise the effects of unfamiliarity with computers and word processors. The subjects used in the pilot study were already familiar with using computers to do word processing. In the full study, subjects who have little or no previous experience will be given the opportunity to practice the required skills in advance of data recording. This will not eliminate all learning effects but should go some way towards reducing their effect.

A related issue is that those subjects who do have computing and word processing experience may use a specialised configuration which affects the behaviour of the keyboard and/or the mouse. For the purposes of this study subjects may only use those options which do not affect the stream of input events reported. Disallowed options include sticky keys and implementation of a key acceptance delay. Subjects used to using these options will again need some time to familiarise themselves with the new behaviour of the computer, and the configuration they are used to will be taken into account when interpreting the data recorded.

Materials

A set of three different tasks was developed and tested in the pilot study. The tasks are based on Apple Macintosh 475 8/160 computers[1], and the ClarisWorks[tm2] word processing package. The computers are equipped with a variety of accessibility hardware and software, including adjustable tables, wrist rests, a variety of different designs of mouse and screen reading and magnification software.

The use of this simple equipment, particularly in finding the most appropriate positions for the monitor, keyboard and mouse, can prevent many performance errors occurring, by making the physical environment as comfortable as possible. It also minimises the effort required to operate the computer which decreases fatigue.

The primary focus of this research is those performance errors that remain even after the physical environment has been optimised, and so providing a comfortable environment for the subjects is essential.

Subjects perform each task in their own time, and are allowed rests between tasks if desired. In the first task, users type out a set passage twice. The first time without any error correction, and the second time with error correction. The former provides easily analysable data, while the latter is more realistic sample of typing, and introduces problems that occur when errors are made in corrections. It also gives some indication of the time spent correcting errors. The passage is constructed so as to test the user's ability to reach all parts of the keyboard, and to use the shift key in conjunction with keys in a variety of positions. It requires a minimum of 54 keypresses, including 25 uses of the *Shift* key.

The second task focuses on use of the mouse, and tests skills such as pointing, clicking, double and triple clicking, and dragging.

The final task is an editing task, which requires use of both

[1] The computers are owned by Lothian Regional Council, and access was kindly provided by the Hands-On-Technology Group at the South Bridge Resource Centre.

[2] ClarisWorks is a registered trademark of Claris Corporation.

the keyboard and the mouse. A pre-typed passage is edited, the editing tasks covering the same set of basic skills as are examined in the typing and mouse tasks. This task is included to allow identification of higher-level patterns associated with typical editing tasks, particularly those involving combinations of keystrokes and mouse movements.

After subjects one and two had completed the study, the mouse and editing tasks were modified. The instructions were simplified, and passages altered to highlight the targets. All scrolling, and other actions that alter the positions of text items on the screen, were moved into the editing test. This means that the positions of the targets in the mouse test are constant, making analysis of pointing actions much easier. The scrolling in the editing test was designed so that all pointing targets would still remain in a constant position, as far as was possible. The tests still cover the same basic set of skills.

Procedure

For each subject, the environment was made as comfortable as possible. Subjects were then allowed time to become familiar with the environment to be used for the test. All terms used in the tasks, such as 'dragging' and 'scroll bar' were explained, and subjects had the opportunity to practice any skills with which they were unfamiliar. For a subject who is easily fatigued, this familiarisation would either be carried out at an earlier date, or a reasonable rest allowed before starting the study.

The tasks were administered by the same observer for each subject. She explained each task as it was presented, and provided verbal help where the subjects required it. The time taken to complete the tasks depends very much on the subject. In the pilot study, subjects took from 30 minutes to two hours to complete all the tasks.

To minimise fatigue, the tasks were presented in the order: typing 1, mouse 1, editing, mouse 2, then typing 2. The text window was of a fixed size and was placed in a standard position on the screen, so that screen co-ordinates can later be related to the objects on the screen. Where required, rests between tasks were allowed. For each subject, the following data was recorded:

- *An automatically generated log of input events*, containing the times of every key press and release, and mouse button press and release. Mouse movements are also recorded. The logging software is described more fully in the next section.
- *A video of the subject performing the tasks.* This is useful in establishing the actual performance errors that occurred.
- *Observations made during the word processing tasks.* For each subject, the same observer recorded impressions and particular examples of the keyboard and mouse difficulties experienced by the subject. These observations are later combined with the video evidence to establish what performance errors actually occurred.
- *Background information about the subject.* This includes the mouse design they were using, previous experience with computers and word processors, the set of configuration options they usually use (if known), and their reported levels of fatigue and ease of performance of the tasks.

AUTOMATIC LOGGING OF INPUT EVENTS

One of the conjectures to be tested by this study is that some performance errors can be recognised by characteristic patterns they produce in the input event stream. Patterns could appear at a number of levels in the input stream. For example, at a high level, difficulties with tasks such as dragging could be indicated by repeated consecutive drags starting from approximately the same position. At the lowest level of detail, accidental additional key presses could be recognised by their timings relative to deliberate key presses. In order to allow investigation of performance errors at all levels, a detailed log of keyboard and mouse events is required.

To provide information about accidental key press timings, and the length of key presses, the log must record the time of every key down and key up event, and must distinguish between events generated by holding down a key, and those generated by repeatedly pressing a key. If difficulties in pressing more than one key simultaneously are to be examined, the same information must also be available for control keys such as *Shift* and *Option*.

Similarly, timings and locations for the mouse down and mouse up events of mouse clicks are also important. The log should record the path taken by the mouse between clicks, so that higher level patterns in mouse movement can be identified. It is then possible to look for correlations between these patterns, and the tasks being performed by the user.

Because of the level of detail required, existing input recording packages were either unsuitable or too slow, and so a fast, unobtrusive logging mechanism – `InputLogger` – was purpose built for this project.

`InputLogger` is a Macintosh specific program which records log data in a file for later analysis. Each line in the log represents an input event, and includes the following information:

```
<time> <eventType> <eventData>
```

The time is measured in *ticks*: sixtieths of a second. The event type could be any one of: key down, key up, key repeat, mouse down, mouse up, or mouse move. For keyboard events, the event data recorded is the hexadecimal ASCII code for the relevant key. Unused ASCII codes are employed to specify keys such as *Shift* which have no ASCII code of their own. For mouse events, the event data is the mouse position on the screen, recorded as separate x and y co-ordinates. A mouse movement is recorded whenever the mouse changes direction by more than 5 degrees. This provides an accurate record of the path taken by the mouse, while reducing the number of mouse positions recorded.

The program is implemented as a system extension, and logging of input events is switched on and off via a control panel. Data is logged regardless of the application being used. The extension operates by trapping all events before they are reported to applications. Events of interest are copied to an internal store managed by Apple's `Audit` library. The control panel then independently reads events from this store and writes the log file.

On Macintosh machines, pressing control keys such as *Shift* or *Option* does not generate a keyboard event, but modifies future keyboard events. Similarly, mouse movements also do not generate Macintosh input events. Instead, the mouse position and status of the control keys are checked by the control panel whenever other events occur, including `null` events. Changes in status are written directly into the log file.

PRELIMINARY RESULTS

The pilot study was performed with four volunteer adult subjects. Subjects one and two have motor disabilities caused by stroke, and type with the right hand only, although subject two can use his left hand to operate the *Shift* key. Subject three has neurological damage causing muscle wastage and spasticity in his hands, and types using several digits on each hand. Subject four has impaired dexterity due to incomplete tetraplegia. He types with his right hand, using his left to operate the *Shift* key.

All four subjects chose to use the standard design of mouse. All were familiar with the use of computers and word processing packages, although not necessarily with the Macintosh and ClarisWorks environments.

Data analysis is still in progress. The remainder of this section describes the analysis of the data and gives some interesting initial observations. A fuller description of the experimental methodology, analysis and the results can be found in [Trewin, 1996].

Analysis

The goal of the analysis is to find performance errors, and input patterns which could indicate performance errors. Until patterns indicating performance errors are found, this task cannot be automated, and so the identification must be done by hand. Errors are found by examining places where the input differs from that dictated by the task. These errors are then categorised – only some of them will be performance errors. The remainder are placed in a single error class. This class includes cognitive errors such as spelling errors, or errors caused by a misunderstanding of the task, and any other errors, including those caused by external events such as the subject being nudged, or distracted in some way.

All errors are found by combining the observations made with the video evidence and the recorded log file, and comparing these with the expected input, according to the task. Before the log file is examined the entries are sorted by their time stamps, and then filtered to transform them into a more readable format.

This filtered log file is then annotated, to mark out the performance errors and other important features. Log files are also annotated with indications of the tasks currently being performed, so that, for example, the log for the second typing task will include an indication of where error corrections start and finish, and mouse task logs will include an indication of the target of a positioning movement, or the type of task currently being performed. The annotation scheme is documented fully in [Trewin, 1996].

The annotated log file is automatically processed to extract summary statistics, and to transform the data into formats appropriate for visualisation and further statistical analysis.

The keyboard and mouse data are currently analysed separately. Analysis of patterns involving both the mouse and the keyboard has yet to be carried out. Such patterns, if they exist, should appear in the editing data. An example of such a pattern would be the use of the mouse to roughly position the cursor, combined with the arrow keys used for fine positioning.

Keyboarding

Table 1 summarises the typing tests for subjects one, three and four. Subject two had no keyboard difficulties, and is excluded from this analysis. The table shows the total number of keypresses made, time taken, total number of performance errors (P) and other errors (O) made, and the time spent correcting errors of each kind. For all subjects, the majority of errors were performance errors, and where errors were corrected, more time was spent correcting performance errors than other errors.

Subject / Test	No. of keys	Time (secs)	Total Errors		Correcting Time (s)	
			P.	O.	P.	O.
1/1	588	929	39	18	0	0
1/2	1348	1844	36	24	334	260
3/1	550	228	49	3	0	0
3/2	637	308	48	7	56	17
4/1	565	276	16	2	0	0
4/2	559	271	7	2	0	0

Table 1: Summary of the typing tests

Non-performance errors included misreading the passage to be copied, forgetting to type some words, and using the *Caps Lock* key as a replacement for *Shift* when trying to type punctuation.

	S1		S3		S4		Total
	1	2	1	2	1	2	
ad	38	34	9	4	4	0	89
db	0	0	30	35	0	0	65
mi	11	6	10	3	2	1	33
dr	0	3	0	1	8	6	18
re	0	0	0	1	2	0	3
tr	0	0	0	2	0	0	2

Table 2: Observed performance errors

The performance errors observed for each subject's two tests,

and total number of errors of each kind are summarised in Table 2. Performance errors are classified as follows:

1. *Additional Key Errors* (ad): A key adjacent to the intended key is activated.
2. *Doubling Errors* (db): An alphanumeric key is unintentionally pressed for longer than the default key repeat delay (16 ticks).
3. *Missing Errors* (mi): The intended key is missed entirely.
4. *Dropping Errors* (dr): The subject fails to press two keys simultaneously (e.g. use of the *Shift* key).
5. *Remote Errors* (re): A key not adjacent to any intended key is pressed (e.g. the subject accidentally leans on a key).
6. *Transposition Errors* (tr): Two keys are transposed.

As can be seen from Table 2, although each subject made performance errors of several different kinds, each individual tends to be prone to one or two specific types.

The most commonly occurring performance error was that of pressing two keys at once, categorised as type 'ad'. Subject one was particularly prone to this, and an example of his typing is shown as the leftmost of the two graphs in Figure 1. For this subject, the timing patterns of accidental additional key presses proved to be highly distinctive, with the unintended key presses overlapping with the intended key press in all 9 instances where the intended key was actually pressed[3], while the deliberate key presses are well separated in time. Intentional overlapping key presses were confined to the use of the *Shift* key and did not occur in his normal typing. Furthermore, in 83% of these cases, the intended key was the first to be raised.

A similar pattern was exhibited by subject four, who also types with one hand and never overlaps keystrokes except when using modifier keys or where 'ad' type errors occur.

These preliminary results are encouraging for the prospects for automatic recognition, and perhaps correction, of this type of keyboard difficulty for those whose normal typing consists of non-overlapping keystrokes. Statistical knowledge-based methods, incorporating knowledge of English digram frequencies or a dictionary, could potentially be used to calculate the intended letter. The positioning of the errors on the keyboard, and direction in which the errors occur, may also provide further material on which to base decisions about these errors.

However, this pattern does not apply to subject three, who types using several digits on each hand. The right hand graph in Figure 1 shows an example of subject three's typical keystroke timings. Many of the keystrokes overlap, and the sample contains only one error of type 'ad'. Since subject three made only 13 such errors, more data is required in

[3] In the other 13 examples of this error, the intended key was missed.

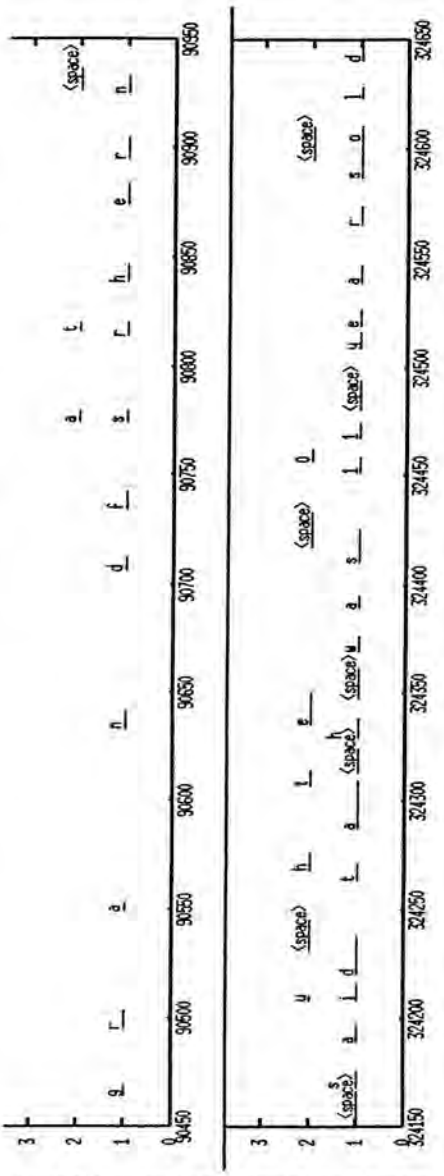


Figure 1: Example keystroke timings (the X-axis represents ticks, the Y-axis separates out simultaneous key presses for presentation purposes)

order to ascertain whether any pattern exists. No patterns based on the keystroke timings are immediately obvious. For those typists who, like subject three, naturally overlap their keypresses, it may be difficult to detect errors of this type.

Doubling errors, caused by pressing keys for too long, were the greatest difficulty for subject 3. This error is extremely easy to detect, given the times of key down and key up events. Here, a doubling error was recorded when a key was pressed for 16 ticks or longer, which is the default key repeat delay on a Macintosh. Such errors can be eliminated by extending the key repeat delay, or deactivating the key repeat facility.

Dropping errors occur when the *Shift* key, or another modifier key, is raised before the key to be modified has been pressed down. These errors were most noticeable in the typing of subject four. 14 dropping errors occurred, and in each of these, the *Shift* key was raised not more than 2 ticks before the modified key was pressed down. In the log files for this subject, wherever the `Shift` key was pressed down alone, this was a dropping error. An example of subject four's typing, showing two dropping errors, is shown as the left graph in Figure 2. For this subject, dropping errors are easily recognisable from the log file data.

Subject one also had difficulty in pressing two key simultaneously, and the low number of dropping errors recorded is due to his use of the `Caps Lock` key as an alternative, even when typing punctuation. An example for subject one is shown as the right hand graph in Figure 2. First, he managed to hold down the key, but then struck the *left* key instead of '?'. After attempting to correct this mistake, he tried again, but dropped the *Shift* key, producing '/' instead of '?'. After correcting this mistake he succeeded in producing '?'.

However, because subject one is used to using the *Sticky Keys* facility, he often pressed the *Shift* key in isolation, forgetting that *Sticky Keys* were disabled for the purposes of the experiment. The pattern observed for subject four, therefore, does not apply to subject one. However, when subject one deliberately pressed the *Shift* key, it was an indication that the next letter was to be upper case, and so despite the fact that no dropping error had occurred, the conclusion that the subject may benefit from using the *Sticky Keys* facility would still be valid.

Other, higher level patterns may also be useful in recognising dropping errors. For example, when typing text, a full stop followed by a space, a *Shift* key press and then a letter is likely to be a dropping error. Further investigation, and more data, is necessary in order to establish whether dropping errors produce reliable patterns in the input stream.

The remaining types of error – remote keypresses and transposition errors – were not observed in large enough numbers to allow examination of their properties at this stage. In the full study, more examples may be observed. Those errors that occur infrequently will not have a large effect on the usability of computer input devices, and so it is less important to find methods of recognising them. It is hoped that the full study will give an impression of which errors occur most frequently, so that effort can be concentrated on finding ways

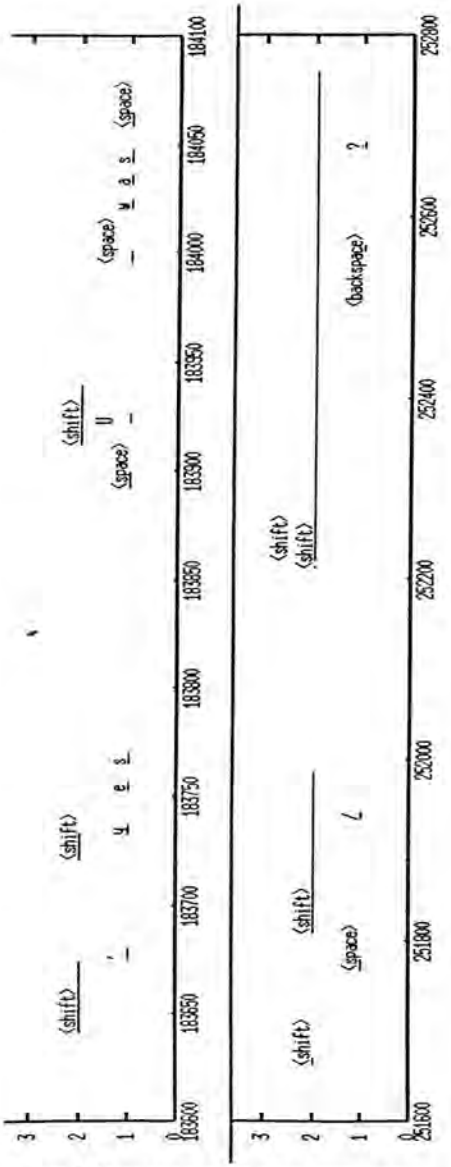


Figure 2: Dropping errors (the X-axis represents ticks, the Y-axis separates out simultaneous key presses for presentation purposes)

of eliminating them.

Mouse Usage

All subjects chose to use the standard mouse design, but found it difficult to use for certain tasks. Dragging and choosing items from hierarchical menus were reported as the most difficult mouse tasks to perform. This supports previously reported findings that dragging is a more difficult task than pointing [MacKenzie *et al.*, 1991]. The performance of subject two improved noticeably the second time the mouse task was performed, despite reporting tiredness. This may be partly explained by some difficulty in understanding the mouse tasks as they were specified, and partly by practice effects. The other subjects did not report tiredness, and their performance remained constant for the two mouse tasks.

The following mouse-based performance errors are recognised:

1. Click length: A click may be too long or too short. Click length is particularly important while using scroll bars.
2. Time between clicks: If this is too long, an intended double click is interpreted as two separate clicks. If it is too short, two clicks are interpreted as a double click.
3. Click movement: If the mouse is not held still within or between clicks, then a the action may be interpreted as a drag, rather than a click.
4. Positioning: The subject does not accurately point to their target.
5. Dragging errors: The subject has difficulty in positioning the mouse with the mouse button held down.
6. Dropping errors: The mouse button is unintentionally released while dragging.

In addition, the path taken by the mouse from a source position to a target is also significant, even though erratic or indirect paths are not explicitly labelled as performance errors by this classification.

One interesting feature to examine is the amount of movement between the mouse down and mouse up events while clicking the mouse. Most applications tolerate a small amount of movement within a click, but if the movement is too large, the click becomes a dragging operation. The Macintosh operating system records a click if the mouse down and mouse up events fall on the same object. Some standard Macintosh targets, such as the close box for a window, are as small as 10 pixels square. In the ClarisWorks word processor, the target area when clicking between the letters 'i' and 'l' is only 3 pixels wide when 18 point type is used. Both of these examples appear in the mouse and/or editing tests.

There are several different kinds of mouse click: single clicks, double clicks, triple and multiple clicks, presses and drags. A *press* is a click which specifies the length of time for which some action occurs. For example, clicking on one of the arrows of a scroll bar is classified as a press.

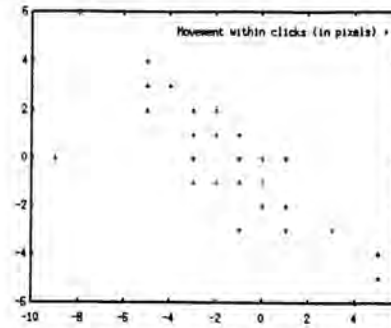


Figure 3: Cursor movement while clicking

Table 3 summarises the number of clicks of each type made by each subject, giving the number in which there was no movement, and the number in which the mouse moved during the click. Dragging operations are omitted, since the movement within them is intentional. Of the four, subjects one and two had the most difficulty in holding the mouse still while clicking. Figure 3 graphs the differences in position between the mouse down and mouse up events for the 71 recorded mouse clicks by subject one. The mouse down position is (0,0) and the mouse up position is shown relative to that, in pixels. 22 of the mouse up positions were at least 3 pixels removed in one direction from the corresponding mouse down position.

Interestingly, the mouse movements made by this subject while clicking tend to follow one diagonal. This tendency could usefully be taken into account by an application accepting clicks from this subject, allowing some leeway along that diagonal.

Clicks	S1	S2	S3	S4
Single (no movement)	16	18	22	26
Single (movement)	39	8	4	5
Double (no movement)	0	2	12	8
Double (movement)	11	5	1	2
Multiple (no movement)	0	1	3	1
Multiple (movement)	4	3	1	3
Presses (no movement)	11	27	15	15
Presses (movement)	25	14	1	3
Total (movement)	79	30	7	13
Total	106	78	59	63

Table 3: Mouse click movement

Analysis of pointing and dragging actions, and click timings is still in progress.

CONCLUSIONS AND FUTURE WORK

The pilot study has provided much useful information on which the full study can build. As the study progressed, the experiment design was assessed, and some modifications

were made to the mouse and editing tasks. Overall, the design proved successful, with the observations and video evidence combining to provide a good impression of what performance errors occurred.

Initially, the mouse and editing task instructions were unclear in some areas, and subjects sometimes had difficulty in finding the targets they were to point to or click on. After modification of the mouse and editing tasks, the situation improved, and subjects three and four required noticeably less assistance in carrying out the tasks. The pilot study also highlighted the need to ensure that enough practice time is available, without fatiguing the subject. Potential fatigue has also been reduced by ordering the tasks so that the typing intensive and mouse intensive tasks are interleaved.

The `InputLogger` software proved effective. The timing accuracy was adequate (although it probably would not be for a fast touch typist), as was the accuracy of recorded control key presses. The logging software is unable to record the movements of the mouse while dragging. This is because the Macintosh does not generate events while the mouse button is held down. This restricts the information available about dragging and selection from menus. However, higher level patterns, such as repeated drags from similar positions, could be used to identify difficulties with dragging.

The preliminary results obtained from this pilot study are encouraging. Although the passage to be typed was short, many examples of six different performance errors have been observed. It is also interesting to note that all of the subjects had very different error profiles, and each was particularly prone to one or two of these error types.

When the full study is completed, the data gathered will provide information about the major difficulties experienced with the manipulation of the keyboard and mouse, and their relative frequencies both within and between individuals. It may also be possible to identify correlations between different performance errors. This information will be useful to software designers and providers of access features for systems and applications.

Patterns in the timing of keystrokes, and in mouse movement while clicking, have already been observed. To test for the presence of characteristic input patterns indicating specific performance errors, more data is required. The data gathered to date indicates that some patterns will be specific to individuals, while others may be globally applicable.

Where such patterns exist, they could be used as a means of identifying difficulties a user is having with the supplied input devices. This information can then be used to choose, and even implement, an appropriate machine/application configuration for a specific user. This would enable provision of automatic support for configuration, thereby improving the accessibility of standard computer input devices and applications.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the support of the University of Edinburgh in funding this research, and the members of the H.O.T. Project and Lothian Regional Council, for providing access to appropriate machines. Thanks also to the volunteer subjects, and to Helen Pain and Mike Ramscar for their time and useful feedback.

REFERENCES

Broadbent, Steven and Curran, Sandra. (1992). *The assessment, disability and technology handbook*. North West Regional ACCESS Centre and Oldham Education Department, Oldham.

Brown, Carl. (May 1992). Assistive technology computers and people with disabilities. *Communications of the A.C.M.*, 35(5):36–45.

CALL Centre. (July 1994). Alternative access to the Apple Macintosh: Using in-built features. University of Edinburgh, 4 Buccleuch Place, Edinburgh.

Karat, John. (1988). Software evaluation methodologies. In Helander, Martin, (ed.), *Handbook of Human-Computer Interaction*, chapter 41, pages 891–903. Elsevier Science Publishers B.V.

Lee, K. S. and Thomas, D. J. (1990). *Control of Computer-based technology for people with physical disabilities: an assessment manual*. University of Toronto Press, Canada.

MacKenzie, Scott, Sellen, Abigail and Buxton, William. (1991). A comparison of input devices in elemental pointing and dragging tasks. In *Proceedings of CHI*, pages 161–166. New York. ACM.

Monk, Andrew. (1987). How and when to collect behavioural data. In Baecker, Ronald M. and Buxton, William A. S., (eds.), *Readings in human-computer interaction : a multidisciplinary approach*, chapter 4, page 138. Morgan Kaufmann, Los Altos, Calif.

Peterson, J. (1980). Computer programs for detecting and correcting spelling errors. *Communications of the ACM*, 23(12):676–687.

Trewin, Shari. (1996). Gathering and analysing keyboard and mouse data. DAI discussion paper, AI Dept, University of Edinburgh, In preparation.

Vassiliou, Yanis, (ed.). (1984). *Human factors and interactive computer systems : proceedings of the NYU symposium ...*, Norwood, N.J. Ablex Publishing Corporation.

Dynamic Modelling of Keyboard Skills: Supporting Users With Motor Disabilities

Shari Trewin* and Helen Pain

Department of Artificial Intelligence, University of Edinburgh, Scotland

Abstract. This paper describes the effective application of user modelling to the assessment of the physical ease with which a user can operate a standard QWERTY keyboard. The application is unusual in the sense that physical rather than cognitive skills are being modelled. The model examines four important skills which a user may have difficulty with, and produces an assessment of the ideal keyboard configuration for that user. This assessment can then be used to adapt the keyboard. For users with motor disabilities, such adaption can minimise or even eliminate the problems they experience. The model dynamically adapts to the current user and operates on free English text input. It has been evaluated using typing data from twenty keyboard users with disabilities and six without. The configuration recommendations made are very well matched to the users' problem areas.

1 Introduction

Computer users with motor disabilities can experience difficulties with the operation of QWERTY keyboards. If we were able to identify and model the specific difficulties of individual users, we could then use such models as the basis for recommendation of a more appropriate keyboard configuration for each user. We believe that this would make the keyboard easier to use, and reduce the number of errors occurring. This paper describes the development and evaluation of techniques for identifying and modelling keyboard difficulties. Our focus is on the modelling of physical skills, rather than on cognitive skills.

Although alternative input devices (such as switches) are available, many users with disabilities find that keyboards provide a more efficient input device. Errors that occur through physical difficulty in manipulating the keyboard are referred to here as *performance errors*. Empirical research with keyboard users with disabilities has highlighted six common classes of performance error (Trewin and Pain, 1996a). These are:

1. *Long Key Press Errors*: An alphanumeric key is unintentionally pressed for longer than the default key repeat delay. On the majority of operating systems, there is a *Repeat Keys* facility, which allows the user to control the length of time a key must be held down for before it repeats. Setting an appropriate delay, or disabling key repeats altogether, can prevent long key press errors.
2. *Dropping Errors*: The user fails to press two keys simultaneously (e.g. use of the *Shift* key). This error type is just one manifestation of difficulty in pressing down two keys at once. The *Sticky Keys* facility, when activated, causes modifier keys to latch. When pressed, they stay active until the next key has been pressed. With *Sticky Keys*, a user presses *Shift* and then 'a' to produce a capital 'A'. Use of *Sticky Keys* can eliminate dropping errors.

The authors acknowledge the support of the University of Edinburgh in funding this research.

The model used by a configuration support application must be dynamic (as defined by Kass and Finin, 1989): it must be able to adjust to the changing requirements of users, which may vary greatly according to factors such as fatigue. It must also adapt to different users who may be using the same computer, where there may be no explicit indication of the change of user. Further requirements are that the model should be unobtrusive and general, so that users are not required to perform specific tasks in order for their keyboard skills to be assessed. Ideally assessment should take place during their normal typing, potentially allowing a large volume of typing data to be examined.

Despite the problem of interest being characterised as a traditional user modelling problem, many of the common techniques used are not suitable here. Those that rely on stereotyping (Rich, 1989) are not applicable: similar keyboard problems may stem from very different disabilities, and similar disabilities may produce very different performance errors.

Approaches using bug libraries and overlay models, such as those used in intelligent tutoring systems (see Clancey, 1987, and Brown and Burton, 1978) are also either too restrictive or inappropriate. These models capture information about how a student's skills and knowledge differ from those of an expert, and are dependent on knowing the user's task, and either identifying missing knowledge or hypothesising about the reason for any incorrect answers. Since free text is permitted, a mechanism reliant on knowing the text a user was trying to type would be too restrictive. For similar reasons, feature based modelling (Webb and Kuzmycz, 1996) is also inappropriate. A further constraint on such approaches is their assumption of consistency in the user's behaviour. Keyboard errors are highly inconsistent, in that they do not occur at every possible opportunity: not every key press will be too long, for example. It is the frequency of errors that indicates those with genuine difficulties. Even in teaching domains, this assumption can cause problems (Self, 1988). Any technique for modelling keyboard skills must deal in frequencies, rather than binary values like known/not known.

The model should also be capable of managing uncertainty over the classification of a character sequence as being correct or containing some performance error. Uncertainty arises here because the user's task is unknown. Established numerical techniques for managing uncertainty – Bayesian networks and Dempster-Schafer theory (Jameson, 1996) – are not ideal. Bayesian networks could in principle be applied, but the full power of this technique is not required, due to the small number of sources of evidence available. Similarly, the ability of Dempster-Schafer theory to combine pieces of uncertain evidence is also not required, as the information sources available are reliable. Given the simplicity of the data available, less complex (and less theoretically motivated) criteria have proved adequate for decision-making.

Because of the uncertainty in the interpretation of an input stream, the model must be tolerant of errors in the performance error recognition mechanisms. It must also be sensitive to medium term variations in the user's typing characteristics, so that the configuration can be altered as the user's requirements change. The following section outlines the approach taken.

3 Recognising Keyboard Difficulties

The model of typing abilities focuses on the four classes of performance error for which some compensatory mechanism exists or has been proposed. Investigation of these areas is carried out unobtrusively by trapping and examining keyboard events before they are passed on to the application in use.

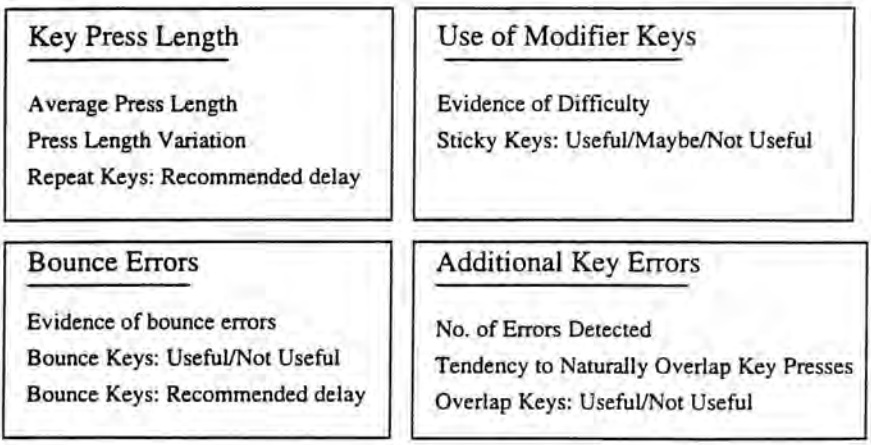


Figure 1. Outline of the user model.

The user model itself is outlined in Figure 1. It contains both general information about the user's typing characteristics and specific information about the recommended keyboard configuration for the current user. The model is dynamically updated as evidence about the current user's typing abilities is gathered. Threshold values and decay of evidence over time are used to damp out the effect of small variations in typing style, and of any errors made in the recognition of difficulties. Note that no changes are made to the actual keyboard configuration in use – the model simply makes recommendations.

Throughout the assessment of keyboard difficulties, an assumption is made that the user is typing English text, probably in a word processing application. The model uses a database storing the frequency with which a given character is followed by another given character in modern English.[1] The frequencies were calculated from the British National Corpus, which contains over 100 million words, representing many different varieties of English. (More information is available at: `http:info.ox.ac.uk/bnc`.) The digram information could be replaced or supplemented with similar statistics about languages other than English, or any command or programming language.

The key repeat delay chosen for the current user is based on their average key press length, and the amount by which their key presses tend to vary upwards from that average value. These calculations are limited to those keys which are rarely deliberately repeated. For example the *Backspace* key is often held down for a long period in order to delete a sequence of characters, and is excluded from the calculation. In addition, abnormally long key presses are ignored, on

[1] The use of digram frequencies, as opposed to a dictionary, has a number of advantages. It eliminates effects due to misspellings in other parts of a word, or words not in the dictionary, and can also handle errors involving the space bar. Digram lookup is also faster than dictionary search, and requires less memory. Speed of classification is important, since the model should not visibly affect the response time of the user's application.

the basis that they are likely to be deliberate, or caused by an event such as the user leaning on the keyboard. The recogniser chooses a value which is longer than approximately 98% of key presses. If large numbers of abnormally long key presses are observed, it is recommended that the repeat facility should be disabled.

Assessment of the user's ease of use of modifier keys is based on the observation that, in the data available, subjects who had difficulty in pressing two keys at once would often type characteristic keystroke sequences, or adopt specific strategies for avoiding multiple key presses. Recognition of difficulties in pressing multiple keys at once is based on the detection of such patterns, and these are weighted according to the strength of the evidence they provide. Indicative patterns include:

- Use of *Caps Lock* for a single key press.
- Pressing a modifier key, followed by a small letter, followed by the *Backspace* key.
- Starting a sentence with a small letter.

In the vast majority of the 163 additional key presses observed in the data available, the unintended key press overlapped in time with that of the intended key. Given this observation, all overlapping keystrokes are candidate additional key errors. Using knowledge of the keyboard layout, English digram frequencies, and the current user's typing style, each overlap is classified as deliberate, an error, or of unknown cause. In the data available, 77% of the subjects rarely or never deliberately overlapped keystrokes, so the user's typing style is an important source of information in this process. The current keyboard layout is that of a Macintosh QWERTY keyboard, but other keyboards could easily be used.

Detection of bounce errors is the most difficult of the four areas tackled by the model. Many people who make bounce errors are also capable of fast deliberate double key presses. The recogniser therefore has two challenges: to spot people who are making bounce errors, and to select a delay which will minimise deliberate key press loss, while eliminating as many bounce errors as possible.

The recogniser operates by examining all double letters and assessing their likelihood of being bounce errors. Knowledge of word processing, English and the timing of the double letter is used. For each double, an evidence value between zero and ten is calculated. The greater the value, the higher the system's confidence that a bounce error has occurred. The choice of value for the delay to be imposed is conservative, preferring to miss some bounce errors rather than eliminate deliberate double key presses.

4 Evaluation

Evaluation of the model is based on typing data gathered from an empirical study of the keyboard difficulties experienced by people with motor disabilities, described by Trewin and Pain (1996a). Twenty subjects with motor disabilities and six without were asked to type a set passage twice. The passage was approximately 100 words (547 characters) long and required 25 uses of a modifier key. The errors made were established through direct observation and video evidence, while a detailed log recorded the `KeyDown` and `KeyUp` events reported to the computer, including timings measured in *ticks* (sixtieths of a second). Macintosh computers and the ClarisWorks word processing package were used.

From this study, 44 log files were available. These were used to simulate direct computer input – reading from the file instead of the event queue.[2] When the whole log file had been read, the state of the user model was examined. For long key press errors, additional key errors and bounce errors, the accuracy of the model's configuration recommendations is assessed by examining the number of errors occurring in each typing test, and where possible comparing this with an estimate of the number of errors that would have occurred had the recommended configuration been used. For modifier key usage, a more sophisticated approach is required, to take into account the coping strategies adopted by users who find it difficult to press two keys at once. Assessment of the model in this area is based not only on error numbers, but also on the user's reported and observed ease of using both hands, and their preference (if known) for using *Sticky Keys*.

The model is text-independent, but because the evaluation data consists of many copies of the same text passage, further evaluation of the modelling techniques in real situations is necessary to increase confidence in the accuracy of the model over general English text.

The following sections describe the results achieved by these techniques in each of the four areas of keyboard difficulty where configuration may be helpful.

5 Detection of Long Key Press Errors

Long key press errors were the most common type of difficulty found in the original study, and choosing an appropriate setting for the key repeat delay is for many the single most important mechanism for improving keyboard usability.

Table 1 shows the results of the model for the twenty subjects with disabilities (1-20) and the comparison group (C1-C6). The table shows each subject's reported level of difficulty in making quick key presses, the repeat delay setting (measured in ticks) recommended by the recogniser for each of their two typing tasks (T1 and T2), the number of long key press errors that would have occurred in each task had the recommended repeat delay been in force for the whole of the time spent typing, and the average key press length.

The recommended delays ranged from 10 to 41 ticks, and the maximum number of long key press errors remaining in a single task was 17, for Subject 17, which represents a 2.8% error rate. In the original data, the maximum error rate for a default repeat delay of 16 ticks was 66.6%, for Subject 13.

Six of the subjects reported some difficulty in making short key presses, and indeed the three subjects for whom the longest repeat delays were suggested were among this group. A total of 17 subjects, including one from the comparison group (a novice computer user), were advised to use key repeat delays longer than the default.

One interesting result is that for Subject 5, who rated short key presses as 'easy'. In both later typing tasks the average key press length was 5 ticks, but the variation among key press lengths was large. She made many long key presses, particularly in the second task (fatigue may have contributed to the difference). Because the recogniser takes into account this variation, long repeat delays are advised in order to cope with the longer key presses she sometimes makes. A

[2] In a real situation, dynamic events rather than a log file would be used. This avoids potential misuse of logs for the assessment of typing productivity.

Table 1. Long key press recognition.

Subject	Reported Difficulty	Setting Chosen		Errors Remaining		Average Key Press Length
		T1	T2	T1	T2	
1	easy	15	18	1	0	7
2	easy	12	-	0	-	5
3	easy	21	21	4	5	9
4	easy	11	11	1	2	4
5	easy	19	30	11	13	6
6	easy	24	22	12	10	10
7	some difficulty	24	25	11	11	11
8	easy	22	-	15	-	12
9	easy	19	21	9	12	9
10	hard	38	-	5	-	17
11	easy	11	11	3	2	4
12	easy	35	32	1	0	16
13	very hard	41	-	0	-	20
14	hard	22	-	1	-	10
15	hard	21	23	2	0	10
16	easy	12	12	0	0	5
17	easy	26	-	17	-	10
18	easy	20	-	0	-	9
19	extreme	34	36	1	0	16
20	easy	24	23	1	1	10
C1	easy	19	-	0	-	8
C2	easy	11	12	0	0	5
C3	easy	11	11	0	0	4
C4	easy	12	12	0	0	5
C5	easy	10	11	0	0	4
C6	easy	13	13	0	0	5

recogniser based purely on average key press lengths would be unable to accommodate subjects with similar wide variations in their key press lengths.

The projected total number of long key press errors for all subjects using their recommended setting is 151, as opposed to the 2610 projected errors under a default key repeat delay. This represents a significantly improved individual configuration for the subjects studied.

6 Detection of Problems Pressing Two Keys at Once

The results of modelling difficulties in the use of modifier keys are summarised in Table 2. The table shows, for each subject,[3] the difficulty they reported in performing multiple key presses, the

[3] The comparison group are included in this analysis but excluded from the table – none had difficulty in using modifier keys, and *Sticky Keys* was not suggested or recommended for any of them. Only one dropping error occurred, and very little evidence of difficulty was gathered.

number of dropping errors they made in each typing task (T1 and T2), the final total of accumulated evidence of a need for *Sticky Keys*, the *Sticky Keys* setting recommended by the recogniser ('on', 'off' or 'maybe'), and the setting that would have been chosen for each subject in a real situation. This last value was arrived at by considering whether the subject is a predominantly one-handed typist, their reported level of difficulty, their preferred configuration, and the number of dropping errors they made.

Table 2. Modifier key difficulty recognition.

Subject	Reported Difficulty	Dropping Errors		Sticky Keys Evidence		Recommended Setting		Ideal Setting
		T1	T2	T1	T2	T1	T2	
1	impossible	0	3	47	81	on	on	on
2	easy	0	-	7	-	off	-	on
3	easy	0	1	0	0	off	off	off
4	moderate	8	6	25	19	maybe	maybe	maybe
5	hard	0	0	1	1	off	off	on
6	impossible	2	3	70	60	on	on	on
7	easy	4	0	22	4	maybe	off	maybe
8	easy	0	-	2	-	off	-	off
9	some difficulty	0	2	3	6	off	off	maybe
10	very hard	0	-	71	-	on	-	on
11	impossible	0	0	84	48	on	on	on
12	some difficulty	4	2	54	53	on	on	on
13	easy	3	-	14	-	maybe	-	maybe
14	easy	0	-	0	-	off	-	off
15	hard	11	2	36	61	on	on	on
16	easy	0	0	9	0	off	off	off
17	easy	0	-	0	-	off	-	off
18	easy	0	-	43	-	on	-	off
19	moderate	0	0	11	0	maybe	off	maybe
20	moderate	2	2	58	14	on	maybe	on

The configuration recommended agrees with the 'ideal' configuration for 22 of the 26 subcts. Of the four cases where the recogniser made a less than ideal choice, three were subjects r whom *Sticky Keys* might be useful, but for whom it was not recommended. All of these subcts were able to use modifier keys competently, but typed predominantly with one hand. The cogniser cannot detect how awkward or tiring an action may be for the user, and can only judge eir actual performance.

In the remaining case, that of Subject 18, the model mistakenly recommended the use of *icky Keys*. This recommendation was based on the observation that Subject 18 used the *Caps ck* key for all single capital letters. This was actually due to a lack of understanding of the yboard, rather than difficulty with modifier key presses. Cases such as this may be common, d the possibility that the user does not understand the use of *Caps Lock* or *Shift* should be nsidered by any system interpreting these recommendations.

Overall, the performance of the model is good. The use of *Sticky Keys* was recommended for all those who rated modifier key presses as very hard or impossible.

7 Detection of Additional Key Errors

Additional key errors were relatively common in the data available, and usually involved two key presses which overlapped in time. Table 3 shows, for each subject, their reported ease of isolating keys to press (Reported Difficulty), whether a problem with additional key errors was found by the model (Problem Indication), and the actual numbers of additional key errors made in the tests (Additional Errors) .

Table 3. Additional key error recognition.

Subject	Reported Difficulty	Problem Indication		Additional Errors		Number Detected		Spurious Errors		Deliberate Overlaps
		T1	T2	T1	T2	T1	T2	T1	T2	
1	moderate	1	1	29	29	23	24	0	2	no
2	easy	0	0	0	-	0	-	0	-	no
3	some difficulty	0	0	5	4	2	2	0	0	yes
4	some difficulty	0	0	3	0	1	0	0	1	no
5	some difficulty	0	0	2	0	2	0	0	0	no
6	easy	0	0	0	1	0	1	0	0	no
7	some difficulty	0	0	3	0	0	0	0	0	no
8	easy	0	0	3	-	1	-	0	-	no
9	easy	0	0	2	0	2	0	0	1	a little
10	easy	0	0	3	-	2	-	0	-	no
11	easy	0	0	3	1	3	0	0	0	no
12	easy	0	0	2	0	0	0	0	0	no
13	easy	0	0	0	-	0	-	0	-	no
14	some difficulty	0	0	0	-	0	-	0	-	no
15	some difficulty	0	0	5	7	3	3	1	0	no
16	easy	0	0	1	0	0	0	0	0	no
17	easy	0	0	4	-	4	-	0	-	no
18	easy	0	0	0	-	0	-	0	-	no
19	moderate	1	0	18	11	8	3	1	2	yes
20	easy	1	1	14	9	10	8	0	0	no
C1	easy	0	-	0	-	0	-	0	-	no
C2	easy	0	0	1	0	0	0	3	2	yes
C3	easy	0	0	0	2	0	0	0	0	a little
C4	easy	0	0	1	0	1	0	1	1	a little
C5	easy	0	0	0	0	0	0	1	0	no
C6	easy	0	0	0	0	0	0	0	0	no

A problem was identified for three subjects. These were the three subjects who made the most additional key errors, including the two who reported the most difficulty. The majority of the remaining subjects did make some additional key errors, but their error rates were low.

The table also shows the number of genuine errors that were detected (Number Detected) – 63% of those present in the original data. Errors are missed most frequently for those subjects who often deliberately overlap keystrokes. These are indicated in the final column of the table (Deliberate Overlaps). For these subjects, particularly Subject 19, additional key errors are difficult to distinguish from normal typing. The error detection mechanism is conservative, in order to avoid detection of errors where none exist. Nevertheless, 16 spurious errors (Spurious Errors) were found, also shown in the table.

To allow for spurious errors, a rate of one or two errors in every 100 characters is tolerated. Error rates above this threshold will cause a problem to be flagged. For some users who make additional key errors, *OverlapKeys* may provide a useful level of support.

Table 4. Bounce error recognition.

Subject	Bounce Key Setting		Bounce Errors		Bounce Evidence	
	T1	T2	T1	T2	T1	T2
1	off	off	0	0	0.0	0.0
2	off	-	0	-	0.0	-
3	off	off	0	0	0.0	0.7
4	off	off	0	0	0.0	0.0
5	off	off	0	0	0.0	0.0
6	2	2	8	12	17.9	6.9
7	off	off	0	0	0.0	2.3
8	off	-	0	-	0.0	-
9	off	3	0	1	0.8	5.8
10	off	-	0	-	0.0	-
11	3	off	0	0	7.1	0.0
12	off	off	2	1	0.0	1.0
13	6	-	3	-	7.9	-
14	off	-	0	-	0.0	-
15	5	off	3	3	16.1	0.0
16	off	off	0	0	1.6	0.0
17	off	-	0	-	2.8	-
18	off	-	0	-	0.0	-
19	off	off	3	0	1.8	0.0
20	5	off	5	3	18.5	3.2
C1	off	-	0	-	0.9	-
C2	off	off	0	0	0.0	2.3
C3	off	off	0	0	0.0	0.0
C4	off	off	0	0	1.6	0.1
C5	off	off	0	0	1.8	0.0
C6	off	off	0	0	0.0	0.0

8 Detection of Bounce Errors

Seven subjects made up the total of 44 bounce errors. The results of bounce error detection are shown in Table 4. The use of *Bounce Keys* is recommended in six of the eleven tasks in which at least one bounce error occurred. Six bounce errors were on the *Caps Lock* key, including all three of the errors in Subject 15's second typing task. Bounce errors on *Caps Lock* cannot be detected by this mechanism, or eliminated by the use of *Bounce Keys*.

Bounce Keys is also recommended for one subject who did not make bounce errors. In this case, a high level of spurious evidence had been gathered. A longer typing test is necessary to reveal whether this evidence would decay into insignificance, or whether similar results would develop for other subjects.

The *Bounce Keys* settings chosen by the model varied between 2 and 5 ticks. Imposing these recommended delays on reactivation of keys would have eliminated 15 of the 38 bounce errors not on the *Caps Lock* key. They would also eliminate 5 deliberate key presses. It is difficult to separate deliberate key presses from bounce errors, and so the results here seem a good compromise – losing one deliberate key press for every three errors eliminated.

9 Summary

We have developed a model of a user's keyboard abilities in four important areas. In all of these areas, existing or proposed keyboard access facilities can alleviate or eliminate difficulties that a user with motor disabilities may experience. The model uses simple statistical techniques. Unlike many traditional user modelling techniques, it has no knowledge of what the user is attempting to type. The solutions are dynamic, user-specific and unobtrusive.

The accuracy of the model has been evaluated using a set of 44 recorded typing logs, made by twenty users with motor disabilities, and six without. Evaluation on dynamic typing data is in progress. If the recommendations made by the model were applied to the original logs, the chosen *Repeat Keys* settings could have reduced the number of long key press errors from 2610 to 151. Use of *Sticky Keys* where recommended could have eliminated 54 of the 56 dropping errors, and would have helped 9 of the 11 subjects who reported difficulty in using modifier keys. The use of *Overlap Keys* was suggested for all 3 subjects prone to additional key errors, and the use of *Bounce Keys* was suggested for 5 of the 7 subjects who made bounce errors. Throughout the four areas and 44 tasks, in only two cases did the model recommend the use of an unnecessary facility. One of these was due to the subject's misunderstanding of the use of modifier keys.

The model we have developed makes explicit configuration recommendations, on which a user is free to act. It does not, however, offer the user any support with understanding, finding and setting the recommended options. While simple recommendations leave the user in control, some users may be unable to alter their configuration themselves. The authors' current work is investigating the feasibility of an adaptive configuration support system incorporating this model, and the use of such a system to actively help keyboard users with motor disabilities to find and set up the keyboard configuration that best suits their needs.

References

Allen, R. (1990). User models: Theory, method and practice. *International Journal of Man-Machine Studies* 32:511–543.

Broadbent, S., and Curran, S. (1992). *The Assessment, Disability and Technology Handbook.* Oldham: North West Regional ACCESS Centre and Oldham Education Department.

Brown, J., and Burton, R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. *Cognitive Science* 2:155–192.

Casali, S. (1995). A physical skills based strategy for choosing an appropriate interface method. In Edwards, A. D. N., ed., *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities.* Cambridge University Press. chapter 17, 315–341.

Clancey, W. (1987). *Knowledge-Based Tutoring: The GUIDON Program.* MIT Press.

Jameson, A. (1996). Numerical uncertainty management in user and student modeling: An overview of systems and issues. *User Modeling and User-Adapted Interaction* 5:193–251.

Cass, R., and Finin, T. (1988). A general user modelling facility. In *Proceedings of Computer Human Interaction*, 145–150. New York: ACM.

Cass, R., and Finin, T. (1989). The role of user models in cooperative interactive systems. *International Journal of Intelligent Systems* 4:81–112.

Rich, E. (1983). Users are individuals: Individualising user models. *International Journal of Man-Machine Studies* 18:199–214.

Rich, E. (1989). Stereotypes and user modeling. In Kobsa, A., and Wahlster, W., eds., *User Models in Dialog Systems.* Berlin: Springer-Verlag. 35–51.

Self, J. (1988). Bypassing the intractable problem of student modelling. In Gauthier, G., and Frasson, C., eds., *Proceedings of Intelligent Tutoring Systems '88*, 18–24.

Smutz, P., Serina, E., and Rempel, D. (1994). A system for evaluating the effect of keyboard design on force, posture, comfort and productivity. *Ergonomics* 37(10):1649–1660.

Trewin, S., and Pain, H. (1996a). Keyboard and mouse errors due to motor disabilities. DAI research paper 838, AI Dept, University of Edinburgh. Submitted for publication.

Trewin, S., and Pain, H. (1996b). On the adequacy and uptake of keyboard access facilities for people with motor disabilities. DAI research paper 839, AI Dept, University of Edinburgh. Submitted for publication.

Webb, G., and Kuzmycz, M. (1996). Feature based modelling: A methodology for producing coherent, consistent, dynamically changing models of agents' competencies. *User Modeling and User-Adapted Interaction* 5:117–150.

A Model of Keyboard Configuration Requirements

Shari Trewin and Helen Pain
University of Edinburgh
Department of Artificial Intelligence
80 South Bridge
Edinburgh, Scotland
44 131 650 2725
shari@dai.ed.ac.uk, helen@dai.ed.ac.uk

ABSTRACT

is paper presents a user model: a computer gram which examines the behaviour of a l computer user. The model encompasses r aspects of keyboard use which can sent difficulties for people with motor abilities. Where relevant keyboard figuration options exist, the model chooses ropriate settings for these options. The del bases its recommendations on ervation of users typing free English text. s intended to form part of a dynamic figuration support tool. Empirical luation showed the model to be very urate in identification of a given user's ficulties. Where recommended figuration options were tried by the ticipants, high levels of error reduction l user satisfaction were found.

Keywords

boards, motor disabilities, empirical studies, user delling, keyboard configuration, Sticky Keys, Repeat s, Bounce Keys.

INTRODUCTION

nputer users with motor disabilities can experience ficulties with the operation of QWERTY keyboards. ough alternative input devices such as switches, and board aids such as keyguards are available, some people that despite their difficulties, bare keyboards provide a er, more efficient or more comfortable input device.

r of the most commonly observed [4, 9, 11, 17] iculties with keyboards are:

- *Long key press errors*: These occur when an alphanumeric key is unintentionally pressed for longer than the default key repeat delay. On the majority of operating systems, there is a *Repeat Keys* facility, which allows the user to control the length of time a key must be held down for before it repeats. Setting an appropriate delay, or disabling key repeats altogether, can prevent long key press errors.
- *Difficulty in using modifier keys*: One handed typists in particular may find it difficult to press two keys at once. The *Sticky Keys* facility, when activated, causes modifier keys to latch. When pressed, they stay active until the next alphanumeric key has been pressed. With *Sticky Keys*, a user presses *Shift* and then 'a' to produce a capital 'A'.
- *Additional key errors*: Some users often press keys adjacent to the intended key. One facility - *Slow Keys* - may be useful for some users who make additional key errors. *Slow Keys* introduces a time period for which a key must be held down before it registers. It would be appropriate for users who press unwanted keys for only a short period of time.
- *Bounce errors*: These occur when the user unintentionally presses a key more than once. These errors are targeted by the *Bounce Keys* facility, available on many operating systems. *Bounce Keys* introduces a delay after a key press, during which time the same key cannot be reactivated. The length of this delay can be adjusted.

In the field of assistive technology, many authors emphasise the need for systems to be configured to suit individual users with disabilities [2, 6, 11, 13, 14]. System configuration facilities such as *Sticky Keys* and *Repeat Keys* can have a dramatic effect on the usability of keyboards by people with motor disabilities [3, 4].

Unfortunately, users are frequently unaware of facilities that could be used to customise their particular environment, and knowledgeable human help is not always available. In the study described here, only 35% of the participants with disabilities had a computer teacher available. The remainder relied on themselves, friends, colleagues and family members for support. It can be difficult for users to assess their own configuration requirements. For example, a user (or indeed an observer) may not know whether two copies of a character appeared because of a long key press error, or a bounce error. Choosing specific settings for the key repeat delay, or the bounce delay, is often a matter of

SETS '98 Marina del Rey CA USA
yright 1998 1-58113-020-1/98/4...$5.00

or the bounce delay, is often a matter of trial and error. To help to bridge the gap between the available facilities and their potential users, some form of automated mechanism may be useful.

As a mechanism for guiding interface adaption, user models could play an important role in improving the accessibility of systems and applications at a number of levels. McMillan [8] suggests incorporating specific models of different subsets of users with special needs into standards for off-the-shelf computing interfaces. User models have also been used in specific assistive technology systems. Accelerated writing systems such as PAL [10] and Messenger [19], record information about a user's frequently used words in order to improve the efficiency of word prediction. Gutkaf, Thies and Domik [5] describe a user model incorporating information about abilities such as colour perception, used to adapt the presentation of colour charts in literature published on the web.

This paper describes the evaluation of techniques for identifying the four major keyboard difficulties described above, and generating a model of the user's requirements for *Sticky Keys*, *Repeat Keys*, and *Bounce Keys*.[1] If successful, such a model could form the basis of an automated configuration support system capable of focusing the interaction on the facilities most relevant to each user, and helping them to choose appropriate settings.

The following section provides an overview of the model itself. The evaluation methodology used, and the participants in the evaluation, are described in Section 4. Some results of the evaluation are presented in Section 5 and discussed in Section 6.

3. OUTLINE OF THE MODEL

The problem of choosing an appropriate keyboard configuration is in many respects a traditional user modelling problem. For example, it fits the definition of "the knowledge and inference mechanism which differentiates the interaction across individuals", suggested by Allen [1]. The system is required to adapt to individual users whose requirements vary enormously, and it takes responsibility for ensuring successful system-user communication (see [12] and [7]).

The model has the following important goals:

- *Dynamic response*. It is envisaged that support for configuration, and therefore a model of keyboard skills, would be most useful on shared or public access machines with a large number of different users, including novice users. In general, the configuration requirements of different users with motor disabilities vary enormously. Individual variation is also possible: one participant in the current study reported that on a bad day she uses *Sticky Keys* on her Macintosh Powerbook, while on a good day she doesn't. In general, variation could occur both between and within sessions of computer use. The model must therefore be capable of responding dynamically to changes within and between users.
- *Unobtrusive operation*. Ideally, users would not be required to perform specific tasks in order for their requirements to be assessed. This produces a more general, flexible model which does not make demands on the user's time and energy, and allows a potentially large volume of data to be examined. This approach has the consequence that the user's task is not known. In our model, the task is not completely unknown: users are assumed to be typing English text in a word processing application. Other languages, including command or programming languages, could easily be substituted.
- *Stability*. In addition to having sensitivity to users' medium and long term changes in requirements, the model must be stable in the absence of changes in the user's style. It is required to handle uncertainty in interpretation of users' inputs, and inconsistency in their manifestation of difficulties, without thrashing between different options.

Model development was informed by typing data gathered in an empirical study of keyboard and mouse difficulties by Trewin and Pain [17]. The data was used in the development of appropriate recognisers, and in the internal (formative) evaluation of the model [18], during which the model parameters were tuned.

The model chooses settings for *Sticky Keys*, *Repeat Keys* and *Bounce Keys*. Recommendations for *Slow Keys* are not made, since Trewin and Pain's original study did not include any participants who matched the profile of a potential *Slow Keys* user. Instead, additional key errors are simply counted.

The model operates unobtrusively by trapping and examining keyboard events before they are passed on to the application in use. It dynamically updates the chosen settings as evidence about the current user's typing abilities is gathered. Threshold values and decay of evidence over time are used to damp out the effect of small variations in typing style, and of any errors made in the recognition of difficulties. Note that no changes are made to the actual keyboard configuration in use - the model simply makes recommendations.

3.1 Long Key Press Errors

The key repeat delay chosen is based on the length of key presses observed. The *Backspace* key, arrow keys and modifier keys are excluded.. Abnormally long key presses are also discarded. The formula ((2 * pressLengthAverage) + 3) is used to calculate an initial delay. If necessary, the resulting value is increased until it is longer than 98% of the key presses observed. Both recent and long term averages are monitored in order to detect and respond to changes over time.

3.2 Difficulty in Using Modifier Keys

When using modifier keys, difficulties are less strongly related to errors. In the data available, those who had difficulty in pressing two keys at once would often type characteristic keystroke sequences, or adopt specific strategies for avoiding multiple key presses. Recognition of difficulties in pressing multiple keys is based on the detection of such patterns, and these are weighted according to the strength of the evidence they provide:

[1] *Slow Keys* is omitted, for reasons explained in Section 3.

- Use of *Caps Lock* for a single key press. (4 points)
- Pressing and releasing *Shift*, followed by a small letter, followed by the *Backspace* key. (4 points)
- Starting a sentence with a small letter. (4 points)
- Pressing and releasing *Shift* without modifying a key. (1 point, or 3 if at the start of a sentence)
- Pressing and releasing another modifier key without modifying a key. (2 points)
- Pressing *Shift* immediately after having pressed and released it. (2 points)

3.3 Additional Key Errors

The model's count of additional key errors is based on the observation that, in many of the additional key presses observed, both the unintended and intended keys were pressed down, and these key presses usually overlapped in time. All overlapping keystrokes are therefore candidate additional key errors. Using knowledge of the keyboard layout, English digram frequencies, and the current user's typing style, each overlap is classified as deliberate, an additional key error, or of unknown cause.

3.4 Bounce Errors

The bounce error recogniser operates by examining all double letters and assessing their likelihood of being bounce errors. This likelihood is given a numerical evidence value between zero and ten, calculated using the length of the gap between the key presses, the lengths of gaps between previous double letter key presses, and the frequency with which the letter is doubled in English.

Evidence is summed, and decays as a function of the number of key presses made. When the total evidence exceeds five points, the recogniser selects a *Bounce Keys* delay intended to eliminate as many bounce errors as possible, while minimising the number of deliberate key presses affected.

4. EVALUATION METHODOLOGY

The data used to develop the model were many copies of the same text passage. A fuller evaluation was necessary to determine whether the parameters of the model, particularly the thresholds, decay rates and evidence weights were appropriate for different English texts, longer periods of typing, and previously unseen users. It was also necessary to judge whether the running model would noticeably affect the response time of a word processing application.

The performance of the model was assessed in an empirical study which compared the model's configuration recommendations with the keyboard errors made by the participants. For *Sticky Keys*, where the difficulty experienced cannot be measured objectively in terms of input errors, the model was assessed by comparison with the participants' opinions of the utility.

The evaluation also investigated the time taken by the model to draw conclusions about a user's typing, and the stability of the model's recommendations over a period of typing. However, discussion of these two aspects of the model's performance is complicated by the fact that the user's typing may itself be changing over time. Due to lack of space, this paper does not report in detail on the stability or speed of response of the model, although an overall impression is given.

4.1 Participants

Thirty participants took part in the evaluation. Twenty had a motor disability affecting their use of the keyboard (Participants D1 to D20), while ten had no motor disability (Participants N21 to N30). Disabilities included muscle control difficulties and spasms (5 people), cerebral palsy (4 people), incomplete tetraplegia (3 people), nervous system damage (3 people), effects due to stroke (3 people), multiple sclerosis (1 person) and rheumatoid arthritis (1 person). The effects of these disabilities on keyboard use included tremor and spasm in the hands and fingers, co-ordination difficulties, loss of dexterity, weakness and pain when pressing keys.

There was no significant difference between the disabled and non-disabled groups in terms of age ($t=-1.669$, $p=0.106$) or experience level ($t=-0.162$, $p=0.872$). Experience was measured as the number of years of daily computer use each participant had had, where daily was considered to be five days a week or more. For participants who used computers less frequently, a daily use figure equivalent to their weekly use was calculated.

Five of the participants with disabilities typed with one hand only, while six typed predominantly with one hand but could use the other hand for modifier key presses. The remaining nine typed with both hands (not touch typing). Of the ten participants with no relevant disability, one typed mainly with one hand, six typed with both hands, and three were novice touch typists.

Nine of the participants (D1, D2, D3, D4, D6, D11, D13, D17 and D19) had previously provided data used during model development and internal evaluation. A '*' will be used to indicate members of this group when referring to individual participants.

4.2 Materials

Four matched text passages were used. Each required 625 keystrokes, and included 21 capital letters and 9 punctuation marks requiring the use of the *Shift* key. A standard form was used to record information about each participant and their session.

Macintosh 475 8/160 and Power Macintosh 6100/66 machines, and the SimpleText word processor were used. The model was used in a mode which produces a log file containing detailed information about the keystrokes made by the participant, and the conclusions drawn by the model.

4.3 Procedure

Experimental sessions were limited to two hours, and extended only if the participant chose to continue. Participants were free to stop or rest at any time. Each session was video recorded (the camera being focused on the keyboard) and logged by the model itself. The same observer (the first author) administered each session.

4.3.1 Modelling each participant

The model was activated. Participants were then asked to copy one of the passages as accurately as possible using the default keyboard configuration. On a Macintosh, the default key repeat delay is 16 ticks

(sixtieths of a second). They were free to make corrections if they wished, or to ignore their errors. A variety of approaches to error correction were anticipated, providing information about the model's response to more natural editing actions.

4.3.2 Trying different configurations

Participants were then asked to copy up to three further passages, each with one of *Sticky Keys*, *Repeat Keys*, or a third experimental utility enabled. Prior to using each utility, it was described to the participants, and if necessary a demonstration was given. Participants were not given the opportunity to practice using the utility, in order to allow recording of the behaviour of novice users.

Constraints on the time available, and the limited stamina of some participants, meant that many did not complete four passages. In choosing the order of activation of the utilities, those which had been recommended by the model or thought useful by the participant themselves were given priority over those which had not been indicated as potentially useful.

The order in which the text passages were presented was varied between participants.

The experimenter activated and deactivated the facilities each time, and informed the participant of what had been changed, and the effect on the keyboard. When *Sticky Keys* was activated, participants were told only that they should generate capital letters by pressing and releasing *Shift* exactly once, and then pressing the desired character, or by using *Caps Lock* if preferred.

After having used *Sticky Keys*, participants were asked to rate the usefulness of the facility. Responses were given on the scale: 'not useful', 'somewhat useful', 'useful', 'very useful', or 'essential'. The responses serve as a subjective measure of the quality of the model's recommendations, in the absence of an objective measure based on input errors.

All participants typed passages in a single session, with breaks between passages, with the exception of Participant D8, who typed two passages on two separate sessions, having become too tired to continue after the initial passage on the first session.

4.4 Analysis

Observations made by the experimenter and video evidence were used to manually annotate the recorded log files, indicating types of errors made and time spent correcting errors of each type.

The annotated log files were then automatically filtered and the Systat statistical package [15] used to perform the analyses. Nonparametric statistics were used, as the variables under examination do not have, or cannot be assumed to have normal distributions.

5. RESULTS

In aspects of typing where difficulties manifest themselves in specific input errors, the model can be assessed according to how well it recognises those errors, and to what extent the suggested new configuration eliminates (or is predicted to eliminate) those errors. This approach has been taken for *Repeat Keys* and *Bounce Keys*, which tackle long key press errors and bounce errors respectively. It is also appropriate for additional key press errors, although in this case no specific configuration recommendation is made. In the case of *Sticky Keys*, however, this approach is not viable. Many users who find modifier keys awkward to use do not make input errors when using them. As a result, the model's choice of potential *Sticky Keys* users is assessed by comparison with the opinions of the users themselves, after having tried the utility. While this subjective measurement is not ideal (discussed in Section 6.2), it is ultimately the user who decides whether to adopt a given configuration option, and so a good correlation between the model and users' opinions would be a useful result.

Use of the model had no noticeable effect on response time of the word processor.

5.1 Repeat Keys

The model recommends a specific key repeat delay, measured to the nearest tick. However, on the Macintosh architecture the only delay settings available are 12, 16, 24 or 40 ticks, or suppression of repeats altogether. For the purposes of evaluation, the model's precise recommendation was therefore transformed into the nearest available setting at or greater than that recommended by the model. For example, if the model recommended a delay of 13 ticks, the setting used would be 16 ticks. An optimal setting is one which eliminates long key press errors while still allowing keys to repeat as quickly as possible.

Ideally, the user would have been asked to type with a key repeat delay setting closest to that suggested by the model, and with higher and lower settings. However, restrictions on the time available for the evaluation sessions meant that only a single text passage could be used. While it would have been possible to split the text passage into three sections of approximately thirty-five words, and use a different repeat delay setting in each section, these sections may have been too short to allow users to adjust to the new delay. Instead, participants typed the whole passage with the recommended delay.

When using the default repeat delay setting, the highest error rate observed was 74.4%, for participant D17*. Participants D3*, D4* and D8 also had error rates greater than 10%. For all participants with no motor disability, error rates were 0.3% or below.

Twenty-three participants tried using the nearest available *Repeat Keys* setting at or above the value recommended by the model. The recommendations made by the model for these participants varied between 10 and 40 ticks. When using the new setting, fourteen of the twenty-three participants made no errors. Of the nine who did make errors using the recommended setting, four had error rates greater than 0.5%, the maximum being 2.3%.

Figure 1 illustrates the relationship between the model's recommended repeat delay and the error rate observed for these twenty-three participants. The vertical axis represents their error rate with the new delay, while the lower scale on the horizontal axis represents the model's recommendation. Grey lines divide the plot according to the delay setting actually used, as indicated at the top of the plot. In six of the nine cases where the new error rate was non-zero, the participants were using the exact setting suggested by the model.

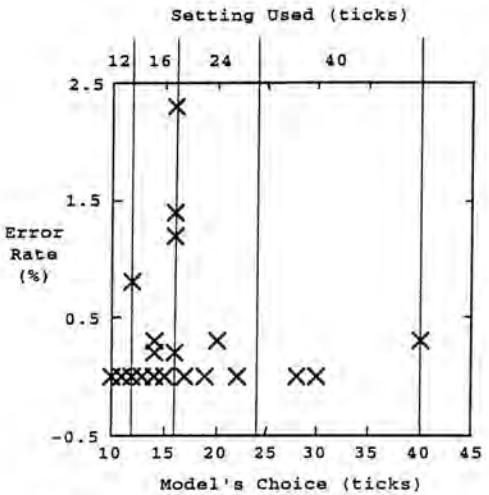


Figure 1: Error rates with altered delay

5.1.1 Stability and responsiveness

In relation to the available Macintosh settings of 12, 16, 24 and 40 ticks, the model's choice of setting was stable for over 99% of the typing of eighteen of the thirty participants. All passages typed were included in this calculation. For all of these eighteen participants, the model stabilised within twenty keystrokes. For other participants, the model's recommendation increased or decreased over time, which may have corresponded with changes in the characteristics of the participants' typing. For three participants the model recommendation did not stabilise, switching between two adjacent settings, or in the case of Participant D2*, between three settings.

Averaging over all participants and all passages typed, the model's recommendation changed every 451 keystrokes.

5.2 Sticky Keys

The model accumulates evidence that the current user may have difficulty in using modifier keys. Initially, the *Sticky Keys* recommendation is 'no'. When the evidence level reaches a threshold of 10 points, the value is set to 'maybe'. When the evidence level reaches or exceeds 30 points, the recommendation given is 'yes'. The evidence value decays in relation to the number of times modifier keys are used.

Twenty-five participants either tried using *Sticky Keys* and then expressed an opinion about it, or had previously used *Sticky Keys*. The evidence value calculated by the model showed significant correlation with the opinions of the participants (Spearman Rho = 0.472, N = 25, $p < 0.05$).

Table 1 illustrates the match between the model's recommendations and the opinions of the twenty-five participants who had tried *Sticky Keys* and expressed some opinion about it. One-handed typists are shown with double underlining, predominantly one-handed typists have single underlining, two-handed typists have dotted underlining, and touch typists have no underlining.

Participant's final opinion	Model's recommendation		
	yes	maybe	no
Essential	D4*		D15
Very useful	D2*, D6*, D12, D16	D7	D5, D14, N28
Useful	D1*	D13*	D10, D19*, D20, N23, N25, N26
Somewhat useful		N22	D3*, N24
Not useful			D11*, D18, N21, N29

Table 1: Sticky Key recommendations

The table shows that all five of the one-handed typists considered *Sticky Keys* at least 'useful', and all were given a 'yes' recommendation by the model. Of the six mainly one-handed typists, five thought the utility at least 'useful', and the model recommended 'maybe' for only two of this group - three potential users were missed. Turning to the two-handed and touch typists, the results initially appear worse: nine of the eleven who rated the utility at least 'somewhat useful' were given 'no' recommendations. However, closer examination of the data gives a different picture. Of these nine false negatives, two considered the utility useful only because it seemed to them to provide a simpler way to generate modified characters. The remaining seven considered the utility useful because there were occasions when they did type with one hand, for example when using the telephone. None had any physical difficulty in using modifier keys in the default way, which is what the model is intended to detect.

5.2.1 Stability and responsiveness

The recommendation made by the model remained stable over all passages typed without *Sticky Keys* for twenty-four of the thirty participants.

Where the model's recommendation was not 'no' at the end of the first passage, the conclusion was reached after the participant had typed between 4 and 21 modified characters, the average being 13.2.

5.3 Additional Key Errors

This analysis examines the accuracy with which the model recognises additional key errors. The number of errors recognised is expected to be less than the total number of errors, since the model cannot identify those errors in which the intended key did not register, or those in which both characters registered, but the key presses did not overlap in time.

Measuring additional key error rate as the number of errors per 100 correctly typed characters, excluding errors

in which the intended key was not activated, and including errors in which the two key presses did not overlap in time, the highest error rate observed was 2.94% (100 observed errors), for Participant D6*. A total of eight participants had error rates greater than 0.5% (D1*, D3*, D5, D6*, D7, D20, N24 and N25).

For the sixteen participants who made more than two errors, the model identified 25% to 75% of the errors. In total, 55% of the errors were detected. The model also sometimes mistook a deliberate letter pair as an additional key error. The maximum number of spurious errors identified for any participant was eight. Over all the participants, twenty-seven had more genuine errors than spurious errors counted by the model, and there was a strong correlation between the total counted by the model, and the actual errors occurring (Spearman Rho = 0.947, $p < 0.01$).

5.4 Bounce Keys

Bounce error numbers are used to identify participants for whom *Bounce Key* may be relevant. The *Bounce Keys* utility was not available for participants to try, so the *Bounce Keys* setting chosen by the model is assessed here in terms of the number of bounce errors it would have eliminated, and the number of deliberate double letters that would have been affected by that setting.

Bounce error rates observed over all the passages typed were very low: twenty-four of the participants made less than one bounce error per 1000 characters. The highest error rate observed was 3.03 errors per 100 correct characters, for Participant D9 (representing 68 actual errors).

The model suggested *Bounce Keys* for the four participants with the highest bounce error rates (0.21 errors per 100 correct characters, or more): D9, D1*, D3* and D17*. During typing of the passages, *Bounce Keys* was also suggested for a fifth participant (D14), but the evidence value had decayed below the threshold by the end of the experiment. *Bounce Keys* was never suggested for any other participant.

The model typically identified 1/3 of a participant's bounce errors correctly, and, on average, wrongly labelled 2 double letters as bounce errors. In no case did the model recommend *Bounce Keys* for a participant who did not make bounce errors.

In the first passage typed, there was a strong positive correlation (Spearman's Rho = 0.766, $p<0.01$) between the evidence value calculated by the model, and the number of bounce errors made by the participants. The correlation decreased over the second and third passages to 0.563 ($p<0.01$), and broke down in the final passage, due to the very small numbers of bounce errors made, and the increasing effect of the initial evidence value at the start of the passage, since the evidence accumulated over all passages typed.

For those participants for whom *Bounce Keys* was suggested, Table 2 shows the delay suggested (two different settings were suggested by the model for D1* and D14), the number of bounce errors they made, and the number of those errors that would have been eliminated had the proposed *Bounce Keys* setting been imposed. The table also shows the number of deliberate double letters typed by each participant, and the number of these that would also have been suppressed by the proposed *Bounce Keys* setting.

Participant	*Bounce Keys* setting chosen	Total bounce errors	Number suppressed by proposed *Bounce Keys* setting	Total double letters	Number affected by proposed *Bounce Keys* setting
D9	3	68	22	178	1
D1*	4/5	11	6/7	96	9
D3*	5	9	4	179	4
D17*	7	3	2	98	5
D14	3/5	1	1	21	0/3

Table 2: Effect of proposed *Bounce Keys* settings.

For all of these participants, many more deliberate double letters than bounce errors were observed. The *Bounce Keys* settings chosen were low, and (with the exception of Participant D14, who made only one bounce error) eliminated 32.4% to 66.7% of the bounce errors observed. The settings would not have affected more than nine deliberate double letters for any participant (including use of the arrow keys, and the *Delete* key), but in three cases (using the larger value recommended for Participant D14) would have affected more deliberate key presses than bounce errors.

5.4.1 Stability and responsiveness

It took between one and seven bounce errors before *Bounce Keys* was recommended for these participants. For participants with low error rates, many characters were typed before this recommendation was made.

The specific setting suggested remained stable for three of the participants, and changed no more than twice for the two others.

6. DISCUSSION

The model was able to identify the participants with the greatest difficulty in all four of the areas studied, and to make good predictions of the configuration that would suit them. The evaluation also showed encouraging results on the stability of the model in general.

However, many of the participants with the greatest difficulty had also provided data for model development: three of the four with the highest long key press error rates; the top additional key error maker; and three of the four with the highest bounce error rates. While the model was accurate in identifying the error tendencies of previously unseen users, it requires further testing on new users with high error rates. Furthermore, three of the five one-handed typists had contributed to the model design. Although the model was equally successful in recommending *Sticky Keys* for these and the two new one-handed typists, further evaluation is necessary in order to test the generality of the techniques used. Further discussion of *Sticky Keys* is given in Section 6.2.

Three novice touch typists were included in the evaluation, while none participated in the original study. The accuracy of the model's recommendations for this

group, is therefore encouraging. The model should also be evaluated by expert touch typists, to examine whether any characteristics of fast touch typing are mistaken for input difficulties.

6.1 Repeat Keys

For participants with high long key press error rates under the default setting, the model's recommendation was clearly effective in reducing these errors.

Some information on the accuracy of the specific setting chosen by the model can be gleaned from the observation that all of the participants with error rates over 0.5% when using the recommended setting were using the exact value chosen by the model. Some of these errors may be attributable to a period of adjustment to the new setting. Nevertheless, these error rates suggest that the model's specific recommendations do not eliminate all errors: a higher recommendation may be an improvement.

The model was quick to settle on a reasonable recommendation, typically stabilising within twenty keystrokes. In general, the stability of the model was also good. Some participants were on the border between two settings. For this group, the model was unstable.

6.2 Sticky Keys

In general, the recommendations made by the model on the use of *Sticky Keys* correlated well with those of the participants themselves after having tried the facility.

The model never recommended *Sticky Keys* for a participant who thought it 'not useful', but failed to recommend it for twelve who did think it at least 'somewhat useful'. However, the majority of this group were two-handed typists who anticipated using the facility only when temporarily typing with one hand. Only three found modifier keys awkward to use, none of whom had provided data for model development. These three were able to accurately use modifier keys in the default way, and could not be identified by the model. There may be input cues not captured by the model which would have allowed identification of these participants. Alternatively, user questioning or human observation may ultimately be a more accurate approach in this aspect of configuration.

In retrospect, participants' opinions were not a good measure for assessment of the model's recommendations - participants should have been asked about their physical ease of use of modifier keys rather than their general opinion of the utility.

The stability of the model was generally good - changes of recommendation were rare. The number of uses of modifier keys made by the participants before the final recommendation was reached was small. The model could therefore identify those participants who make errors in the use of modifier keys fairly quickly, depending on the number of modified characters in the text they were typing.

6.3 Additional Key Errors

The strong correlation between additional key error rates and the model's count of additional key errors shows that despite the uncertainty associated with classification of overlapping adjacent keystrokes the model is accurate enough to be useful in identifying those users who are prone to such errors. Some of these users may benefit from utilities like *Slow Keys*, or from using a keyguard to reduce additional key errors.

6.4 Bounce Keys

The accuracy and stability of the model over the periods of typing observed during evaluation were good. The threshold was effective in eliminating noise due to inaccuracies in discrimination between bounce errors and deliberately typed double letters.

Due to the low bounce error rates observed during evaluation, the model typically observed a long period of typing before ascertaining a need for *Bounce Keys*, in comparison with the period required for *Repeat Keys* or *Sticky Keys*. The model recommended the use of *Bounce Keys* consistently for the four participants with the highest bounce error rates. However, three of these highest rates were still relatively small, and the delays recommended in some cases would have affected more deliberate key presses than errors. In practice, it is possible that only the participant with the highest error rate would have benefited from using *Bounce Keys*. Empirical research is required in order to identify the range of error rates for which *Bounce Keys* is typically useful. This information could be taken into account in a future implementation of the model, and the threshold value adjusted accordingly.

Given that for three of the participants, the specific delay chosen meant that more double letters were affected than errors eliminated, it may be desirable to reduce the recommended delay. However, this would also reduce the number of errors eliminated. Without specific research on the effect of imposing different delay values for users of *Bounce Keys*, it is not possible to evaluate the model's specific delay recommendations in terms of likely user satisfaction. In terms of error reduction, balanced against the requirement to minimize the number of deliberate double letters that would have been affected by the proposed setting, the model's recommendations appear reasonable.

7. SUMMARY

This paper has presented a user model which identifies keyboard users who are having difficulty with the default keyboard configuration, and recommends an appropriate alternative configuration.

The evaluation of the model involved twenty participants with motor disabilities and ten with no motor disability. For both groups, the model was effective in recognizing evidence of difficulty, and did not make any irrelevant positive recommendations. The major limitation of the model is its failure to recognize some potential *Sticky Keys* users. Nine participants may have occasionally benefited from using the utility, but had no difficulty in using modifier keys in the ordinary way during the model evaluation. This group illustrate the point that configuration utilities are useful to a broader range of users than their original target group. However, since the goal of this model is to identify users experiencing physical difficulty in using the keyboard, their exclusion is unsurprising. Of greater concern is the model's failure to recognize the genuine difficulties of three participants who found modifier keys awkward to use. This aspect of the model requires further research in order to establish

whether these results could be improved, or whether this is a hard limitation of this approach.

The model responded quickly to input errors and other evidence that the user was having difficulty, but also showed good stability over typing periods of up to approximately 500 words.

The identification of configuration requirements from free typing is, in the main, a feasible proposition, at least in the context of the aspects of keyboard use studied here. This result suggests that it may be possible to provide users with automated, user-adapted support for keyboard configuration. There is already a lack of awareness of the existing facilities among those who would potentially benefit from them. As the number of available access facilities grows, and the number of options increases, configuration will become an increasingly daunting yet essential task. A system with the ability to identify relevant facilities has the potential to raise users' awareness without overloading them with irrelevant information. The model presented here could form the basis of an efficient system which would help users to quickly find and set a good configuration which will reduce input errors, and make the keyboard easier to use.

8. ACKNOWLEDGMENTS

The authors would like to thank the University of Edinburgh for funding this research; all the volunteer participants and those who helped in contacting participants, particularly Annalu Waller, the Herald and Post, and the Thistle Foundation; Lothian Regional Council and the Hands on Technology project for providing access to much of the equipment used; and Simon Kelly and Mike Ramscar for their assistance and helpful feedback.

9. REFERENCES

[1] Allen, R. User models: Theory, method and practice. *International Journal of Man-Machine Studies*, 32 (1990), 511-543.

[2] Baecker, R. and Buxton, W. Eds. *Readings in Human-Computer Interaction: A Multidisciplinary Approach*. Morgan Kaufmann, Los Altos, California. 1987.

[3] Borden, P. *User's Guide for AccessDOS*. Version 1.0. The Trace Research and Development Center, Madison, WI. 1991.

[4] Brown, C. Assistive technology computers and people with disabilities. *Communications of the A.C.M.* 35, 5 (1992), 36-45.

[5] Gutkat, B., Thies, S. and Domik, G. A user-adaptive chart editing system based on user modeling and critiquing. In *User Modeling: Proceedings of the 6th International Conference*, A. Jameson, C. Paris and C. Tasso, Eds. Springer Wein, New York, (1997), 159-170.

[6] Hansen, P Krogh and Wanner, J. Software drivers for pointers used by persons with disabilities. In *Resna '93: Proceedings of the 16th Annual Conference*. Resna Press, Washington D.C. (1993) 443-445.

[7] Kass, R and Finin, T. A general user modelling facility. In *Proceedings of Computer Human Interaction*, New York. ACM, (1988) 145-150.

[8] McMillan, W. Computing for users with special needs and models of computer-human interaction. In *Proceedings of Computer Human Interaction*, New York. ACM, (1992) 143-148.

[9] Millar, S. and Nisbet, P. *Accelerated Writing for People with Disabilities*. CALL Centre and Scottish Office Education Department, CALL Centre, University of Edinburgh, ISBN 1 898042 01 2, 1993.

[10] Newell, A., Arnott, J., Cairns, A., Ricketts, I. and Gregor, P. Intelligent systems for speech and language impaired people: A portfolio of research. In *Extra-Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*, A. Edwards, Ed. Cambridge University Press. (1995), 83-101.

[11] Poulson, D., Ashby, M. and Richardson, S. Eds *Userfit: A practical handbook on user-centred design for Assistive Technology*. European Commission Brussels-Luxembourg, TIDE 1062 USER project 1996.

[12] Rich, E. Users are individuals: Individualising user models. *International Journal of Man-Machine Studies*, 18 (1983), 199-214.

[13] Shein, F., McDougall, J., Knysh, B., Sainani, D. Lee, K., Brownlow, N. and Milner, M. A model for alternate access systems. In *Resna '89: Proceedings of the 12th Annual Conference*. Resna Press, Louisiana (1989) 17-18.

[14] Shein, F., McDougall, J., Knysh, B., Sainani, D. Lee, K., Brownlow, N. and Milner, M. Guidelines for alternate access system developers. In *Resna '89 Proceedings of the 12th Annual Conference*. Resna Press, Louisiana (1989) 19-20.

[15] SYSTAT, Inc. *SYSTAT: Statistics, Version 5, Edition*. SYSTAT, Inc. Evanston, IL. 1992.

[16] Thimbleby, H. Treat people like computers designing usable systems for special people. In *Extra Ordinary Human-Computer Interaction: Interfaces for Users with Disabilities*, A. Edwards, Ed. Cambridge University Press. (1995), 283-295.

[17] Trewin, S. and Pain, H. Keyboard and mouse errors due to motor disabilities. To be published in *International Journal of Human-Computer Studies*.

[18] Trewin, S. and Pain, H. Dynamic Modelling of Keyboard Skills: Supporting Users with Motor Disabilities. In *User Modeling: Proceedings of the 6th International Conference*, A. Jameson, C. Paris and C. Tasso, Eds. Springer Wein, New York. (1997), 135-146.

[19] Venkatagiri, H. Efficiency of lexical prediction as communication acceleration technique. *Augmentative and Alternative Communication*, 9 (1993), 161-167.

InputLogger: General purpose logging of keyboard and mouse events on an Apple Macintosh

Shari Trewin

University of Edinburgh

Department of Artificial Intelligence

80 South Bridge

Edinburgh

EH1 1HN

Scotland

Phone: 44 131 650 2725

Email: *shari@dai.ed.ac.uk*

ABSTRACT

Event logging, particularly logging of event timing information, is often used in human-computer interaction research in investigations of the ways in which people use computers, and in the evaluation of input devices and applications. This paper describes *InputLogger*: a low-level input event recorder for the Apple Macintosh†. It differs from other keystroke loggers in that it records accurate timing information for all keyboard and mouse events, while being application independent. It is capable of logging any Macintosh session. *InputLogger* has been used to gather data on the difficulties experienced by people with motor disabilities using ordinary keyboards and mice. It would be appropriate for many other experimental applications.

Behavior Research Methods, Instruments, & Computers. **30**(2), 327-331, May, 1988.

Input Logging

Timed input event logging is appropriate wherever computers are used to investigate human behaviour involving time, such as reacting to stimuli, making precise movements, or interacting with a computer application or input device. Timing and/or log information is used in many research areas, including evaluating the usability of different pointing devices (Kabbash *et al.*, 1993; MacKenzie *et al.*, 1991; Riviere & Thakor, 1996), the evaluation of keyboard designs (Smutz *et al.*, 1994), investigating theories of human movement control (Gillan *et al.*, 1990; Walker *et al.*, 1993), looking at individual differences in typing characteristics (Leggett *et al.*, 1991), and evaluation of application programs and their interfaces (Brewster *et al.*, 1994; Roberts & Moran, 1983).

Although the majority of studies are limited to either the mouse or the keyboard, in some areas, such as application interface evaluation, or the evaluation of a whole computer setup, it is necessary to look at the interaction between mouse and keyboard usage. Examples include the work of Douglas and others, who looked at the significance of the time needed to move between the mouse and the keyboard (Douglas & Mithal, 1994), and the author's own investigations into the usability of standard input devices for people with motor disabilities (Trewin, 1996).

While some experimenters have used manual timing and logging mechanisms (Roberts & Moran, 1983), this has generally been due to the lack of availability of general purpose logging mechanisms. Roberts and Moran noted that they used stopwatch timings because: "not everyone has access to an instrumented editor or videotape setup". Where available, an automatic recording is superior in accuracy and detail to a human recording, particularly where timing information is concerned. Automatic logging also frees the experimenter to make observations of events of particular interest, such as errors made by subjects.

Unfortunately, the majority of existing packages for recording Macintosh input events are limited to either the keyboard or the mouse, and many are built into a specific application. This is because they are often custom-written for a specific purpose, such as recording the process of editing a document, or the time taken to point to a given target.

Because of the time and effort required to develop logging software, custom-built loggers tend to record the minimum amount of information necessary for the specific experiment for which they are developed. As a result, they are not as reusable as might be hoped.

The *InputLogger* program attempts to provide a reusable, general purpose logging mechanism for the Macintosh, by recording detailed information about both mouse and keyboard input events. This detail includes key down and key up times for all keys, including control keys, mouse down and mouse up times, and a description of the path taken by the mouse during pointing movements. These logs can then be tailored to suit specific experiments by providing an appropriate filter.

As an introduction, the following section describes the aspects of the Macintosh architecture relevant to recording events. The paper goes on to give a review of existing Macintosh logging software, followed by an overview of *InputLogger*. The benefits and costs of application independence are discussed. The following section describes the implementation of *InputLogger*, covering some of the difficulties in providing accurate logging information on a Macintosh, and how they have been tackled. Those interested in trying the software are referred to the final section, which gives details of its availability.

Macintosh Input Events

Before describing Macintosh logging software, it is worth reviewing the Macintosh architecture, and the way input events are represented internally.

When using a keyboard or mouse, there are a limited number of inputs that a user can make. The Macintosh operating system recognises and processes the following set of input events:

- `KeyDown`: an alphanumeric key has been pressed down.
- `KeyUp`: an alphanumeric key has been raised.
- `KeyRepeat`: an alphanumeric key is being held down, generate another instance of the appropriate character.
- `MouseDown`: the mouse button has been pressed down.
- `MouseUp`: the mouse button has been raised.

`KeyUp` events are often suppressed or ignored, as the majority of applications do not require them.

A time is associated with each event, using the system clock measure of the number of ticks since system startup. A tick is a sixtieth of a second. In addition, a key code is associated with keyboard events.

Mouse movements do not generate events, but the current mouse position is recorded with every internal event, including `NULL` events. This allows the mouse path to be tracked by checking the mouse position for every internal event.

In addition, pressing of modifier keys such as *Shift* or *Command* does not cause any keyboard events. Instead, internal flags are set. These modify the key code reported for other key presses, so that each alphanumeric key is modified in different ways by the flag settings in place when it was pressed down. This treatment of modifier key presses makes it difficult to record accurate information about them. The approach taken in *InputLogger* is described later in this paper.

The *InputLogger* program aims to record as accurate a picture of input device usage as possible. This means logging all of these input events, plus mouse paths and presses of modifier keys.

Existing Macintosh Logging Software

Several tools for recording keystrokes and/or mouse movements for the Macintosh already exist.

For example, JEdit (Eklundh & Kollberg, 1996) is a dedicated text editing application, which records those input events relevant to the text editing process: key down and mouse up. Key repeat events are recorded as key down events. It produces log files with timings of keystrokes to within 0.1 sec. Positions of mouse clicks and selections are recorded as positions within the text, rather than screen positions which would be more difficult to interpret. A mechanism for playing back the recordings made by JEdit is also available.

Another logging text editor is TeachText+, developed by Steve Lang of Loughborough University of Technology, UK (provided by author). It is an extension of the TeachText editor which records timings of key down and mouse down events, to the nearest tick. Mouse clicks,

double clicks and drags are recorded simply as 'mouse events', with a start and end position related to the file being edited.

ProtoTymer (Miller & Stone, 1990) is a system to collect keystrokes and mouse events from a Hypercard application. It allows playback of these events, but imposes severe restrictions on the amount of data that can be stored.

An alternative approach to logging is to make a recording of the screen, which can be played back. This is exemplified by MediaTracks, the Farallon screen recorder. This approach can produce very large and unwieldy log files. In MediaTracks, timing information can only be included by putting a clock on the screen, so accurate split-second timing is not possible.

More general event trappers do exist. Now Utilities provides facilities for recording the use of applications, but is limited to keystrokes only, and provides no timing information. MacroMaker is part of the standard Apple distribution, and will record short sequences of keystrokes and mouse clicks to be replayed on demand. It is, however, only suitable for short event sequences, as it has a noticeable effect on application response times. Timing information can be included but causes very large log files to be produced.

The majority of these loggers are concerned with recording events that affect the system, rather than the user's original actions. Most do not distinguish between key presses and key repeats, or keep a record of mouse movements. Key up events are generally not significant, and some systems omit mouse down or mouse up events, even when the mouse is being dragged. Many systems provide a video or other mechanism for playing back events in the order they occurred, but timing information is often crude, and again the mouse path is generally omitted. The author is not aware of any system which provides full information about the mouse and keyboard usage. The *InputLogger* program is intended to fill this gap.

InputLogger

InputLogger is a system extension/control panel combination which, once installed, generates ASCII log files recording the input presented over a given period. Logging is switched on and off via the control panel. When actively logging events, *InputLogger* records events regardless of what application is in use, and users may switch between applications as often as desired.

Because *InputLogger* is implemented at a low level in the system software, it is fast and unobtrusive. The use of separate processes for reading and writing events allows event records to be stored up during busy periods, such as when the user is typing fast, and written to the log file during a pause. This helps to minimise the effect of logging on the response time of the running application.

The events recorded in the log file are exactly those recognised by the Macintosh operating system: `KeyDown`, `KeyUp`, `KeyRepeat`, `MouseDown`, and `MouseUp`, with the addition of one new event type: `MouseMove`. A `MouseMove` event is recorded whenever the mouse changes direction by more than 10 degrees. This provides an accurate yet compact record of the mouse path taken between mouse clicks, including small-scale shake and large scale positioning movements. The default value of 10 degrees can be altered on the *InputLogger* control panel. For reasons explained in the description of the logging mechanism itself, `MouseMove` events are not recorded while the mouse button is held down. Presses of modifier keys are recorded as `KeyDown`/`KeyUp` pairs, in the same way as other key presses. `KeyRepeat` events are not generated for modifier keys.

The default behaviour of *InputLogger* is to record `KeyUp` events. An alternative version suppresses these events.

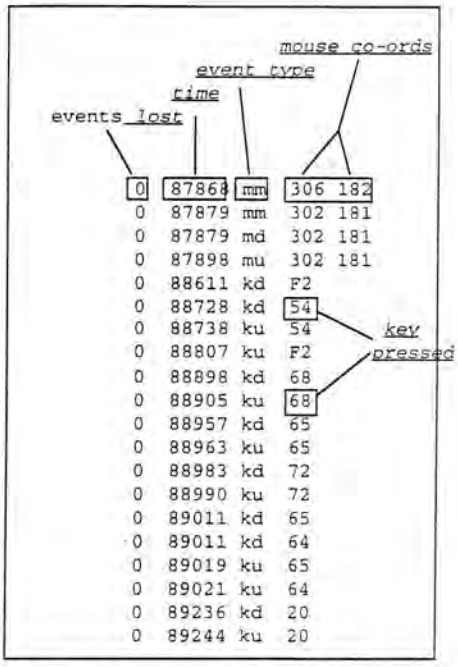


Figure 1: *InputLogger* Example Log File

An example log file is shown in Figure 1. Each log file entry is of the form:

```
<nLost> <time> <eventType> <eventData>
```

- `<nLost>` - is a check for internal logging failures, which should remain zero (see the description of the logging mechanism for more details).
- `<time>` - the time at which the event occurred. Time is measured as the number of *ticks* since system startup. A tick is a sixtieth of a second.
- `<eventType>` - specifies what event has occurred. One of: `KeyDown (kd)`, `KeyUp (ku)`, `KeyRepeat (kh)`, `MouseDown (md)`, `MouseUp (mu)`, `MouseMove (mm)`.
- `<eventData>` - For keyboard events, the event data recorded is the hexadecimal ASCII code for the relevant key. Unused ASCII codes are employed to specify keys such as *Shift* which have no ASCII code of their own. For mouse events, event data is the mouse position on the screen, recorded as separate x and y co-ordinates.

InputLogger was developed in Ansi C using the Metrowerks' CodeWarrior™ development system (Metrowerks Inc., 1993) on a PowerMacintosh 6100/66. The default configuration has been used on both this machine and a Macintosh 475 8/160 with no noticeable performance

degradation. On the PowerMacintosh 6100/66, *InputLogger* has proved adequate for recording 5 minutes of continuous random character typing, including KeyUp events, at a rate of 150 words per minute. It should therefore be adequate for even the fastest touch typist.

The size of the log files generated in a given session depends upon the amount of input generated during the session, rather than the length of the session. One minute of continuous mouse movement will generate a log file of approximately 20k, while 1000 typed characters will be recorded in a 46k log file, assuming KeyUp events are recorded. Only 23k would be required if the option to ignore KeyUp events were used. Should the logging mechanism be used for several hours of keyboard or mouse activity, the upper limit on the size of the log file produced is dependent on the memory available: *InputLogger* imposes no limit.

Application Independence

The use of an application-independent logging mechanism has several advantages over loggers that are built into a custom-designed environment. The ability to log the use of any word processor, graphics package or other Macintosh application provides the experimenter with freedom to study any existing application without the need to modify it, or to design tests using any package or combination of packages.

Using the same mechanism to log different types of task facilitates reliable between-task comparisons. It also allows analysis of use of the operating system itself. Actual computer usage in a real work environment can be recorded directly without interrupting the user or requiring them to run special versions of any software.

The major disadvantage of general purpose logging mechanisms is a technical limitation imposed by the Macintosh operating system. This makes it impossible to record the path taken by the mouse while dragging. For mouse drag path recording, an application-specific logging mechanism would be necessary. *InputLogger* simply records the times and positions of the mouse down and mouse up events of a drag.

It is sometimes also useful to incorporate application-specific information into a log, so that input events can be viewed at a higher level, such as a click on a specific button. It would be possible to extend *InputLogger* to incorporate facilities for higher level event recording, but this would slow the logger. A more efficient solution is to use post-processing of the log files to produce a higher-level trace.

The Logging Mechanism

InputLogger consists of a control panel and a system extension. The extension is a patch on the SystemEvent internal Macintosh routine. It traps keyboard and mouse events of interest and records them. The control panel provides an on-off mechanism, writes events into the log file, and also directly records events of interest that the system extension could not catch.

The control panel and system extension communicate using Apple's Audit© library. This provides an unobtrusive event tracing facility, usable from all types of Macintosh code segments. Events can be written into a store and read from the store. In *InputLogger*, events are written by the system extension, and read by the control panel. *InputLogger* can set the size of Audit's internal store and the behaviour of the library when the store becomes full. If the internal store is full, *InputLogger* overwrites the oldest event in the store. Audit warns whenever events have been lost, and *InputLogger* writes the number of events lost into the log file. If events are being lost, non-zero values will appear in the first column of the log file (see Figure 1). The Audit library does not allow dynamic adjustment of the size of the internal store. If events are lost, the solution is to restart the program with a larger internal store by

changing a constant in the header file and recompiling. While inconvenient, this is unlikely to be necessary, since the default used by *InputLogger* (64 entries) has proved adequate for recording typing rates of 150 words per minute.

Logging alphanumeric keystrokes and mouse events

When logging is switched on, the control panel opens a new log file and enables the Audit record. While the Audit record is enabled, the system extension examines all events reported to the `SystemEvent` routine. Whenever an alphanumeric key is pressed, a `KeyDown` event is generated. The system extension puts the event, including the time at which it happened, and the key that was pressed, into the Audit record. The control panel reads the event from Audit, along with a count of how many events, if any, have been lost, and writes this information into the log file. When the key is released, a `KeyUp` event occurs and is treated in the same way. `MouseDown` and `MouseUp` events are recorded along with a time, and the mouse position for the event.

Logging mouse movements

Mouse movements do not cause input events, but the current mouse position is recorded along with every Macintosh event that is reported to `SystemEvent`. This allows the system extension to keep track of mouse movement by comparing the mouse position of the current and previous event. Writing out a `MouseMove` event every time the mouse position changed would produce extremely large log files, and so *InputLogger* explicitly records a mouse movement only when the direction in which the mouse is moving changes by at least the number of degrees specified on the control panel. Then, the mouse position recorded *before* the change of angle is written to the log file, and the angle of movement between that position and the current mouse position is stored as the new direction of movement.

This, however, is not enough, as the system extension can only look at the mouse position whenever some interface event occurs. An accurate record of mouse movement requires the mouse position to be checked while no interface events are occurring. When no events are occurring, the Macintosh generates `NULL` events. These have the same format as normal events, including a record of the mouse position, but are treated differently. Most importantly, they are not reported to the `SystemEvent` function, and so are not trapped by *InputLogger*'s system extension. Instead, the control panel catches these events and tracks the mouse position in the same way as the system extension does, inserting `MouseMove` events whenever necessary.

Logging modifier key presses

As mentioned previously, modifier keys such as *Shift* or *Command* must be treated differently to alphanumeric keys, as no event is generated when they are pressed. A similar mechanism to that used for mouse movements is employed - every time an event is reported to the system extension, the status of the modifier keys is checked. Every time a change has occurred the appropriate `KeyDown` or `KeyUp` event is lodged with Audit. As with mouse movements, modifier keys must be checked on `NULL` events by the control panel itself. This method provides as accurate a picture of modifier key presses as possible, although it is still less accurate than recording of alphanumeric key presses.

There is one key that cannot be recorded in the same way as the others: the *Caps Lock* key. This key is similar to other modifier keys in that no events are generated when it is pressed. Instead, its status is recorded in internal structures and can be read. However, because it is a locking key, when it is pressed, the appropriate status bit is set, and this bit remains set until the key is pressed again. The length of time for which the key was actually pressed down each time cannot be established by the logging software. Instead, the log shows a single `KeyDown` or `KeyUp` event for each press of the key.

Log file idiosyncrasies and limitations

Events written into the log are generated from two different sources: the control panel and the system extension. These both keep a separate record of the mouse movements and modifier key positions. As a result, sometimes mouse events are written to the file out of order, and modifier key movements are occasionally duplicated. Both the control panel and the system extension use the same event time-stamp mechanism, and so a simple sort of the log files is sufficient to provide a correctly ordered set of events. Duplicated events can be ignored.

With regard to recording mouse drag movements, *InputLogger* is limited by the behaviour of the Macintosh environment. When the mouse button is pressed down, no events are processed until the mouse is released -- if a key is pressed during this period it will not appear on the screen until the mouse is released.[1] Even `NULL` events are not reported. This means that *InputLogger* cannot gain control of the machine in order to check the status of modifier keys and the mouse position until the mouse button is released. Consequently, the path taken by the mouse while dragging, and changes in modifier key status while dragging, cannot be determined.

Potential Extension Conflicts

The use of Macintosh system extensions, while providing a very efficient logging mechanism, also introduces the potential for conflicts between *InputLogger* and other system extensions. Any extension using the Gestalt selector 'klog' will clash with *InputLogger*. Similarly, any program writing to a file with the same path name as the log file may also clash. No other specific clashes are known, tail patching is not used, and other instances of the Audit library will not cause problems. A Gestalt selector or file name conflict can be avoided by changing the Gestalt selector or file name used by *InputLogger*, recompiling and reinstalling, or by disabling the extension which clashes.

Availability

Interested readers can obtain a copy of *InputLogger* from the author, who would welcome feedback and comments. The software, including source code, can be downloaded via ftp from ftp.dai.ed.ac.uk/pub/user/shari/InputLogger.sit.hqx, or accessed via the author's web page at http://www.dai.ed.ac.uk/students/shari, and can also be made available on disk. For the disk version, a fee of £5 (US $10) is requested to cover postage, packaging and time.

† Metrowerks is a registered trademark of Metrowerks Inc. CodeWarrior is a trademark of Metrowerks Inc. The Audit library is copyright 1992-3 Apple Computer Inc. Apple is a registered trademark of Apple Computer Inc. Macintosh is a trademark of Apple Computer Inc.

[1] The highlighting of selected areas during a mouse drag is performed by the Macintosh operating system function: `TEClick` (Apple Computer Inc., 1985). `TEClick` returns only when the mouse button is released, so patching this function would not help. There is a mechanism by which applications can register a function to be called periodically by `TEClick`, but there is no way for *InputLogger* to access this facility for a given application.

References

APPLE COMPUTER INC. (1985). *Inside Macintosh*, **1**, Reading, MA: Addison Wesley.

BREWSTER, S., WRIGHT, P., & EDWARDS, A. (1994). The design and evaluation of an auditory-enhanced scrollbar. *Proceedings of CHI*, 173-179, New York, ACM.

DOUGLAS, S., & MITHAL, K. (1994). The effect of reducing homing time on the speed of a finger-controlled isometric pointing device. *Proceedings of CHI*, 411-416, New York, ACM.

EKLUNDH, K., & KOLLBERG, P. (1996). Computer tools for tracing the writing process: from keystroke records to s-notation. In RIJLAARSDAM, G., COUZIJN, M., & VAN DEN BERGH, H. (eds), *Theories, Models and Methodology in Writing Research*, 526-541, Amsterdam University Press, Amsterdam.

GILLAN, D., HOLDEN, K., ADAM, S., RUDISILL, M., & MAGEE, L. (1990). How does Fitt's law fit pointing and dragging? *Proceedings of CHI*, 227-234, New Yord, ACM.

KABBASH, P., MACKENZIE, I., & BUXTON, W. (1993). Human performance using computer devices in the preferred and non-preferred hands. *Proceedings of INTERCHI*, 474-481, New York, ACM.

LEGGET, J., WILLIAMS, G., USNICK, M., & LONGNECKER, M. (1991). Dynamic identity verification via keystroke characteristics. *International Journal of Man-Machine Studies*, **35**, 859-870.

MACKENZIE, S., SELLEN, A., & BUXTON, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. *Proceedings of CHI*, 161-166, New York, ACM.

METROWERKS INC. (1993). *CodeWarrior 6*. [computer program] The Trimex Building, Route 11, Mooers, NY 12958, USA.

MILLER, D., & STONE, A. (1990) ProtoTymer: human performance instrumentation for Hypercard prototyping. *Proceedings of CHI*, 466.

RIVIERE, C., & THAKOR, N. (1996). Effects of age and disability on tracking tasks with a computer mouse: Accuracy and linearity. *Journal of Rehabilitation Research and Development*, **33(1)**, 6-15.

ROBERTS, T., & MORAN, T. (1983). The evaluation of text editors: methodology and empirical results. *Communications of the ACM*, **26**, 265-283.

SMUTZ, P., SERINA, E., & REMPEL, D. (1994). A system for evaluating the effect of keyboard design on force, posture, comfort and productivity. *Ergonomics*, **37(10)**, 1649-1660.

TREWIN, S. (1996) A study of input device manipulation difficulties. *Proceedings of the Second Annual ACM Conference on Assistive Technologies*, 15-22, USA, ACM.

WALKER, N., MEYER, D., & SMELCER, J. (1993). Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction. *Human Factors*, **35**, 431-458