The ACCESS Framework
reinforcement learning for accessibility and cognitive support for older adults

The ACCESS Framework

reinforcement learning for accessibility and cognitive support for older adults

Michael James Heron

2011

University of Dundee

The ACCESS Framework: Reinforcement Learning for Accessibility and Cognitive Support for Older Adults

A Thesis Submitted for the Degree of PhD at the University of Dundee

Michael James Heron

Declaration by the Candidate

I declare that I am the author of this thesis; that, unless otherwise stated, all references cited have been consulted by me; that the work of which the thesis is a record has been done by me, and that it has not been previously accepted for a higher degree.

Michael Heron

April 2011

Declaration by the Supervisor

I declare that Michael Heron has satisfied all the terms and conditions of the regulations made under Ordinances 12 and 39; and has completed the required 9 terms of research to qualify in submitting this thesis in application for the degree of Doctor of Philosophy.

Professor Ian Ricketts

April 2011

Abstract

This dissertation focuses on the ACCESS Framework which is an open source software framework designed to address four issues with regards to older and novice users with accessibility needs – that they often do not know what support is available within their systems, that they often do not know how to change those settings they know exist, that they often lack the confidence to make the changes they know how to make, and are often unable to physically enable accessibility support.

The software discussed in this dissertation serves as a bridge between what users are **expected** to know and what they **actually** know by assuming the responsibility for identifying user accessibility requirements and making those changes on the user's behalf. User interaction with the framework is limited to either expressing approval or disapproval with regards to corrective action. Individual corrections are deployed as plug-ins within this tool.

Four studies were conducted during this research. Three of these studies were aimed at evaluating the ACCESS Framework directly with the remaining study being an exploration of a cognitive support tool deployed using the framework. Two of these studies involved participants attempting to perform specific, well-defined tasks on systems that had been configured to the extremes of what was possible with operating system settings. These tasks were attempted with and without the support of the framework. The final study was a focus group in which issues of the framework were discussed by individuals who had been through the experimental trials.

The research provided strong evidence that this is an effective mechanism for accessibility configuration when there is a strong match between identified accessibility needs and available operating system support. The system was seen as understandable, useful and appropriate by participants, with a majority stating that they would be willing to use a similar system on their own machines.

Acknowledgements

No postgraduate research is done in isolation, and I would like to express my thanks and acknowledgements to the following people and organisations.

For their tremendous guidance and tolerance during the supervision of this PhD, I would like to express my sincere gratitude to Norman Alm, Peter Gregor, Ian Ricketts and Vicki Hanson. Half of these individuals retired during the lifespan of this project, and while they profess otherwise, I have my suspicions it was simply easier than continuing to supervise me.

Thanks go to IBM for their provision of the grant that funded this research. Without their contribution, in a very real sense this research would never have been conducted in the way it was. Whether that is a good thing or a bad thing is left as an exercise for the reader.

Thanks also go to the SiDE hub at Dundee University for making available the participants for the largest of the studies in this research. Special thanks go to Marianne Dee at SiDE for her tireless patience in shepherding me through the recruitment process and correcting the many organisational mistakes I made as I went along.

Thanks and acknowledgements go to my friends, both within and without the University, both near and far, especially those who have to listen to me explain, often in depth and without stopping to draw breath, topics in which they have no interest.

Lastly, thanks go to my family for their support, and my girlfriend Pauline for her tireless patience, her constant encouragement, and her baffling willingness to put up with me on a daily basis.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 The Aging Workforce

The research outlined in this dissertation addresses the issue of aging in the workplace, with a specific emphasis on developing open source tools aimed at addressing this increasingly important demographic change. That the workforce is aging is a fact that is attested to by countless reports and papers, and that it is having a profound impact on our conceptions of work and retirement is shown by the increasing focus this issue is receiving in political debate. At the time of writing, several countries have announced changes to their long-standing retirement ages in response to the pressures exerted on existing pension and medical provisions.

In the United Kingdom, the coalition government has outlined plans for the state pension age for men to rise to 66 by 2016, a full ten years earlier than had been planned by previous Conservative proposals. The retirement age similarly is planned to increase to 66 for women by the year 2020. The possibility of the retirement age rising to 70 by the year 2046 is still under consideration, but already these plans are causing concern amongst certain sectors of the community (http://www.bbc.co.uk/news/uk-11980214).

In France, the retirement age was raised from 60 to 62 in November of 2010, which provoked considerable public discord and protests from citizens. In Italy, the cabinet agreed to raise the retirement age for women in the public sector to 65 from 60. In Spain, Prime Minister Zapatero has stirred up controversy by announcing plans to raise the retirement age by two years to 67. In Ireland, retirement ages have been increased by a year, and in Germany they have been increased by two.

The issue remains highly controversial, but is being considered by many as the inevitable solution to the issues caused by rising life expectancy and lower potential support ratios.

The end result of this legislation is that individuals will have to work longer, and this has additional implications for organisations that employ older workers. This research then was funded to investigate some of these issues and produce, where possible, open source tools that could ease the transition and maximise the benefit of retaining older workers for longer. The benefits of having individuals working for longer, as far as organisations are considered, can be considerable - particularly in incidences where the experience of those workers is valuable or potentially irreplaceable.

The trend of computerization in the workplace is now well-established, and this presents particular problems for older workers who may be encountering computer systems for the first time in that context. The aging process negatively impacts on many cognitive processes including parsing of visual information, parsing of auditory information, and generalised measures of intelligence such as working memory. All of these issues have implications for the way computer systems are designed and presented at home and in the workplace. Many tools and much research already exist to support older individuals working with computers. The tools, many of which are bundled for free with modern operating systems, are effective and widely available - however, knowing that they exist, and additionally knowing how to activate them - is knowledge that is not necessarily possessed by many computer novices.

This dissertation focuses on the ACCESS Framework, which is a piece of plug-in enabled software that is designed to address some of the issues associated with an aging workforce.

1.2 Terminology

Several terms used within this dissertation have multiple meanings, and this section of the dissertation defines how they are used within the context of the work.

1.2.1 Accessibility Software

Within the context of this dissertation, the phrase 'accessibility software' is used to refer to any software program which is primarily designed to make it easier for individuals to use computers. These may include tools that alleviate physical problems (such as screen readers or magnifiers), and those that reduce the cognitive burden on users when dealing with complex systems.

1.2.2 Framework

Within the context of this dissertation, the word 'Framework' is used to refer to a software framework as is defined in chapter five. A framework is a piece of software which is designed to represent an abstract activity, with specific requirements being represented by supplemental code that is interpreted by the framework. The ACCESS Framework then is the generalisation of a pattern of user interaction and operating system integration.

1.2.3 Plug-In

Within the ACCESS Framework, individual accessibility corrections are handled by plug-ins. The framework itself has no accessibility support built into it, it is a container into which are inserted the plug-ins that implement corrective algorithms. Plug-ins are packages of extra external code that are included within a software application when it runs. They are not part of the code that makes up the application, but the application must be designed in a special way to support them. Plug-ins commonly extend functionality, add extra features, or change existing features of a software package.

1.2.4 Correction

A correction within this dissertation is defined as any action taken by the framework on the user's behalf as a result of analysed user input.

1.2.5 Conditioning / Reinforcement

Within this dissertation, the words 'conditioning' and 'reinforcement' are used to refer to any time the user positively reinforces or positively punishes a plug-in through the 'like' and 'dislike' buttons.

1.3 Structure of Dissertation

Chapter two begins with a discussion of the implications related to an aging workforce with regards to increasing pressures on employers and state support mechanisms such as pensions. In addition, this thesis will discuss the nature of older workers within organisational contexts, their preferences for knowledge work, and the effect on organisations when individuals retire. In this chapter, this dissertation attempt to summarise the existing literature with regards to the issue of substantially shifting demographics, and present a case that these shifts are unprecedented, and have considerable impact on likely future working patterns.

Chapter three contains an overview of the current state of cognitive and accessibility support for older adults, as well as a discussion of why such support is required with regards to the impact of the aging process. In addition, the dissertation will discuss some of the consequences of modern software development with regards to the suitability of the produced software with reference to older users. The chapter concludes with a roundup of some of the more noteworthy software research developments that have been discussed in the literature.

Chapters two and three comprise the literature review of this study - no novel arguments are made within these chapters, they serve primarily to define the context of the discussions that follow.

The nature of the open source development process has implications for software design, and some of these implications are particularly relevant to accessibility tools, especially those aimed at a demographic group that contains a disproportionate number of novice computer users. Accessibility software carries with it an implied (and sometimes explicit) duty of care. There is a burden on the developer to ensure that they have taken what steps they reasonably can to ensure their software is developed and deployed responsibly.

Chapter four then discusses the open source development process with reference to older adults in particular. First the dissertation will discuss why people contribute to open source projects, and then discuss how that contribution manifests itself. Both of these discussions are required to provide the necessary context for the sections on the ethical implications of providing such software to older users. This chapter puts forward some guidelines for how such software should be developed for older users in line with the analyzed literature. While this chapter is primarily an analysis of existing research, it serves as a novel synthesis of the issues – there is, as far as the literature review for the topic can demonstrate, no research on the implications of open source software being developed for potentially vulnerable novice users. All software produced as part of this research is open source – however, it is the contention of this thesis that there remain issues to be addressed in the design of open source tools in order to make them suitable for deployment within the destination demographics.

Chapter five introduces the ACCESS Framework, an open source platform for the easy development and deployment of accessibility and cognitive support tools. This framework is heavily influenced by Trewin's (2000) insight that modern operating systems make it disproportionately difficult to enable accessibility support, and that even knowledge regarding the existence of such support is often not possessed by novices. The framework incorporates an inversion of the traditional mechanism for operating system configuration, and utilizes a positive reinforcement and positive punishment model to allow individuals to express support or displeasure with the changes that are made to their operating context. In addition, it supports a plug-in design that allows for the easy incorporation of new functionality. In this chapter, the dissertation will discuss the design of the tool and the problems it is designed to address. In chapter six, the dissertation will focus on the technical issues that the framework has been designed to resolve, and discuss the specific plug-ins that were written for the research in the dissertation.

Chapter seven introduces a single plug-in of this framework – a context sensitive electronic 'post-it' note. The note shifts its contents based on the current working application of the user. This chapter incorporates a discussion of the plug-in, its academic rationale, and also the results of a small six-person qualitative study conducted to assess the worth of such a tool in a general context.

Chapter eight introduces the experimental design of the two main studies of the research – two thirty-eight participant, quantitative studies into the key mechanism of the ACCESS Framework – its reinforcement learning system. The studies involve the deployment of accessibility plug-ins and a series of mouse manipulation tasks performed on a computer configured to be as unfriendly an operating environment as realistically possible. This chapter also provides the overview of the studies and the hypotheses linked to each.

Chapters nine, ten and eleven discuss the results of this study with reference to profiling the group, assessing the impact of the plug-ins on objective and subjective measure of performance, and analyzing participant feedback on the design of the tool itself.

Chapter twelve discusses the qualitative feedback obtained from a ten participant, three-hour focus group aimed at discussing some of the issues that the experimental data from the studies above did not capture. It also provided an opportunity to investigate issues tangential to the study, with a group of individuals who had two hours of exposure to the tool. This study was broken into three separate sessions. The first hour was devoted to a review of the tool. The second hour was a collaborative, iterative design exercise to explore the design space mapped out by participants during the experiments. The third session focused on open source as a mechanism for delivering software of this nature to older users.

It is the contention of this thesis that accessibility software is unreasonably difficult to develop as a result of the often low-level interactions with the operating system required to make changes. Coupled to this, a number of individuals working within the field of accessibility are not, by training, software developers. As such, the ACCESS Framework has been designed to ensure that accessibility support can be implemented easily by even novice software developers. Chapter thirteen then outlines some of the future research directions linked to the tool - this dissertation focuses on the tool's suitability for end-users, but a second intended user-base is that of accessibility researchers. In this chapter, the dissertation will outline some of the problems associated with academic research software, and highlight ways in which this framework helps address them. Beyond the author’s subjective, and highly biased, experience with the framework, no original research is presented in this chapter. It serves instead as an overview of worthwhile research to follow.

Chapter fourteen concludes the dissertation and discusses the specific contributions to knowledge made with regards to experimental evidence.

1.4 Contributions to Knowledge

It is the contention of this author that this dissertation contains the following main contributions to knowledge, in order of significance:

- An evaluation of the suitability of reinforcement learning as a mechanism for providing user accessibility support within a particular operating system context. Chapters seven, eight and nine cover the evaluation of this concept.
- The provision of a novel open source framework that lowers the barrier of participation in related kinds of accessibility research, as well as providing a platform for other researchers to easily explore the design space mapped out by this dissertation. This research has shown that this is an effective mechanism in certain circumstances, and that it has a high level of support amongst users with regards to perception of its benefit, its appropriateness, and their willingness to use a similar system on their own computers.

- An evaluation of the validity of several specific accessibility adaptations. These are also discussed in chapters seven, eight, and nine:
  - Dynamically adaptive double-click adjustment
  - Dynamically adaptive pointer size adjustment
  - Dynamically adaptive mouse trails deployment
  - Dynamically adaptive mouse speed adjustment
  - Dynamically adaptive pointer precision deployment
- An analysis of the validity of context sensitive electronic notes when individuals are performing well defined, relatively novel activities within familiar computer applications. This contribution to knowledge is discussed in chapter five, where the study is outlined and evaluated.

There are several other minor contributions to knowledge, such as a novel synthesis of the literature on open source development, but since these remain experimentally untested they are not listed as primary contributions above. Similarly, while the framework offers a potential solution to many of the issues discussed in chapter eleven, these also have not been experimentally tested and likewise the tool's utility in this area is not included as a primary contribution to knowledge.

1.5 Research Problem

In the course of this research, each of the studies undertaken has its own set of hypotheses related to the individual tasks being undertaken. However, for the larger central research question, the following main issues are put forward for discussion. These cover the primary intended audience for the tool - end users.

Being able to make adjustments to an operating context is first predicated on knowing what can be changed within a computer system. Where an individual lacks this knowledge, they are unable to investigate the issue and turn that awareness into knowledge or action. While every year more older users become more familiar with what they can do with their computer systems, it is one of the hypotheses of this research that the majority of participants in the trial will self-identify themselves as lacking the knowledge of the context of what the possibilities are within the configuration options of their systems. As part of this research then, it will be necessary to identify within the context of the study as to whether a system like that which is proposed in the dissertation is needed, or will be perceived to be valuable by users.

In cases where older users know what can be changed, the next issue to be met is knowing how to make the change. The culture of jargon associated with computer systems, and the often complex workflows required to perform even simple operations both work to make it difficult for novice users to make the changes to their system. An individual may know that, for example, their mouse cursor moves too fast, and they may know that this is a thing that can be changed. That does not necessarily translate into knowing where to look or how to make the adjustments. It is one the hypotheses of this research that the majority of participants in the trial will self-identify a knowledge deficit with regards to making configuration changes to their systems.

Even when an individual knows what can be changed, and understands how to make the change, the unfriendly jargon and complexity of undoing changes can result in users being unwilling to make the change. This is often because they have little confidence in their ability to assess the impact of the change and also to be able to correct mistakes. It is one of the hypotheses of this research that the majority of participants in the trial will self-identify a lack of confidence with regards to experimenting with their computers and making configuration adjustments.

In order for the tool to be considered useful, it must show benefits that would justify its installation on a system. It must be able to pick up the need for corrections, to make corrections that objectively improve user performance, and also make corrections that increase the user's subjective sense of satisfaction with their system. In order for the framework to make meaningful corrections for a user, it must first pick up on the need for corrective actions. While plug-ins triggering will not prove that the plug-in is correctly picking up genuine accessibility concerns, it will show that a plug-in can pick up on the need. In order for the Framework to demonstrate its value it must be seen, at least subjectively, as useful by the participants in the study.

Since each of the plug-ins addressed in chapter eight is aimed at improving performance within a specific category of interaction, it follows that when corrections are made by the framework they will logically result in improvements in objective measures of performance across the relevant actions. It is the contention of this thesis that the tool provided will engineer measurable improvement in objective performance metrics in the experimental tasks.

Independent of actual, objectively verifiable improvements in performance measures, there is an important element of subjective feedback with regards to perceptions of improvement. It is the contention of this thesis that the mechanism of reinforcement learning will lead to improvements in the subjective ratings provided by participants in chapters nine and ten.

Providing a tool that fills a user requirement is only one part of the challenge of introducing it to the desktop - many tools that have real benefits simply are not used because individuals choose not to do so. In order for the tool to be considered a suitable candidate for resolving user problems, it must be something that individuals feel comfortable with, can understand, and appreciate the benefit. This is an issue that will be addressed in the subjective feedback analysed in chapter eleven.

It is the argument of this thesis that existing methods of user configuration of software systems can be replaced with the aggregation of binary data points collected by the ACCESS Framework as a result of its input analysis routines. While manually setting options in the control panel of an operating system offers greater granularity over the fine details of interaction, the complexity and jargon of these options often makes them inaccessible to novice computer users. It is the contention of the thesis that inverting the responsibility for making changes so that it resides with the computer, and limiting the user decision making to either 'I approve', or 'I disapprove', will resolve the issues associated with the knowledge burden of configuration. It is also the contention of the thesis that adjustment of internal weightings and thresholds will allow sufficient granularity to permit a set of plug-ins to meet an individual's particular accessibility needs.

The abstract concept of training the framework on the basis of simple binary choices is anticipated to be a tractable one. However, in order for the tool to meaningfully bridge the gap between assumptions of user capabilities and actual user capabilities, the tool will need to demonstrate that participants understand the impact of their actions, and can assess the implications of their decisions.

Leaving aside subjective and objective measures of performance, an important personal consideration is how comfortable individuals feel with a tool they are using. Previous research into agent based configuration has suggested that the inversion of control associated with this technique can be frustrating and unwelcome, and in order for a tool to be sufficiently valuable to users despite any theoretical benefits, it must be a system with which individuals will willingly engage.

It is anticipated that users will express a theoretical preference to have more control over their system than is provided by the framework, but it is also anticipated that the benefit of the tool will be such that individuals will be willing to use it regardless.

2 The Aging Workforce

2.1 Introduction

We live in a society that is rapidly aging. Certainly in the industrialised West, we are faced with two powerful demographic influences. According to the United Nations' Population Division (2008):

- Life expectancy in developed nations is rising
- Fertility rates in developed nations are falling

Population growth within the United Kingdom for example shows that while the overall population growth sits at eight percent, this has been distributed unevenly across the population groups. According to UK population estimates (Office for National Statistics 2007) in 2007 the population of over-65s grew by 31 percent, and the population of under-16s declined by 19 percent with the fastest growing age group being those aged 80+. A population distribution that was initially triangular is becoming increasingly rectangular. This trend towards an aging population is not unique to the United Kingdom. It has been shown to hold true in Japan (Statistics Bureau, 2008), Italy (United Nations, 2008), Australia (Australian Bureau of Statistics 2005), Canada (Statistics Canada, 2006) and the United States (U.S. Census Bureau, 2005), amongst others.

These trends are concerning enough, but they coincide with a third demographical trend in many nations – the transition of large numbers of the baby boomer generation from the productive workforce into retirement. (Dohm, 2000; DeLong, 2007; Hanson and Lesser, 2009). From 2012, an estimated 10,000 extra people a day will be eligible to draw their pensions (Bone, 2007). The Baby Boomer generation has been referred to as the 'pig in the python' (Jones, 1980), a 'bulge' that transitions through each discrete stage of the population pyramid. It has been said that the sheer size of this demographic group distorts all the life stages through which it passes as a result. The transition of the Baby Boomer generation into retirement has an increased impact due to the number of individuals involved, and the relatively short window of time within which they will transition (Shaw and Smith, 2003).

The combination of these factors will lead to increasing pressures on the formal and informal social support networks that exist to support the elderly in society. Much of the financial entitlements of the elderly are predicated on a certain ratio of younger workers to retirees. As the population pyramids of the industrialized nations change, this ratio can no longer be assumed and alterations to the nature and provision of entitlements will have to follow.

Other sectors of society are also feeling the pressure, such as in the provision of health-care (Centers for Disease Control and Prevention, 2007), the distribution of governmental financial support (Weil,

2006) and replenishment of the workforce generally (Hanson and Lesser, 2009; Peterson and Spiker, 2005). The impact of an aging population is likely to be felt throughout society as we evolve to accommodate these long-term changes in our demographic makeup. These changes will be especially notable in the area of employment – while many Baby Boomers are eligible to retire, there is an increasing trend for individuals to work into their retirement period (Butrica, Schaner, and Zedlewsi, 2006; Pitt-Catsouphes and Smyer, 2005), and an increasing need for governments and employers to support older individuals in the workplace.

2.2 Retirement Age and the Aging Population

The Potential Support Ratio (PSR) is a simple formula that describes the strength of the relationship between adults of working age and those their financial contributions are supporting. The PSR thus describes the financial burden assumed by the working population of a country to support the non-working elderly population. At the beginning of the 20th century, the ratio in the United Kingdom was 13.3. In 1950, it was 6.2. In 1995 it was 4.09.

While it is not possible to predict with certainty how this ratio will be affected in the next few decades, several United Nations predictions suggest that the likely PSR in the United Kingdom will fall to somewhere between 2.3 and 3.0 given current retirement ages and immigration figures (United Nations, 2001).

In the event no changes are made to retirement ages, the United Nations summarizes the situation this way:

> *The current system of providing income and health services for older persons who are no longer working has been based, by and large, on an age structure with a potential support ratio of 4 to 5 persons in working-age for each older person aged 65 years or older. If the current age at retirement does not change, the PSR is projected to decline to about 2.*
>
> *A decline of the PSR from 4 or 5 to 2, or even to 3, would certainly create the need to reconsider seriously the modalities of the present system of pensions and health care for the elderly*. (United Nations, 2001)

Among the suggestions for ways to compensate for a falling PSR, the UN suggests one approach is to increase the age of retirement. This allows for a simultaneous means of both reducing the number of dependents and increasing the number of supporters in the workforce. Their most optimistic realistic scenario for the United Kingdom (PSR of no less than 3.0) requires an increase of the retirement age to 68.2, and to sustain the current PSR would require an increase to 72. In Scotland alone, the number of people of working age is projected to fall by 7% from 3.18 million in 2004 to 2.96 million in 2031.

The dependency ratio in Scotland is expected to fall from 5:3 (five people of working age to support three retirees) to 5:3.5 (Scotland's Futures Forum, 2007).

This figure it should be noted was estimated before recent financial events and the increased public debt associated with these created additional pressure on governments. In Britain, government plans are to increase the retirement age from 65 to 68 by 2046, but consultancies have argued that these measures do not go far enough in light of current economic circumstances (PriceWaterHouseCoopers, 2010), and that a rise to 70 is required to meet the minimum needs of a shifting demographic and financial landscape.

The likely outcome then is that people are going to need to work longer before they can retire. This meshes with research suggesting that the majority of elderly workers would be interested in remaining in employment until seventy years of age or further (Lee, Czaja and Sharit, 2009).

However, as people approach retirement age, their preferred pattern of employment tends to shift. On the whole, older workers prefer knowledge work over physical work (Hanson and Lesser, 2009), and they desire more flexible working arrangements to allow them to more easily balance their work-life and their home-life (Hanson and Lesser, 2009; Platman and Taylor, 2004). Organisations too have started to realize the benefits of having more older workers on the payroll, including less absenteeism, higher morale, and higher retention (Hanson and Lesser, 2009). Research also suggests that managers profess no bias against older workers in certain industries (Furunes, and Mykletun, 2005; McMullin and Tomchick, 2004; Peterson and Spiker, 2005) and feel that sufficient training opportunities are provided, although older workers in certain contexts have reported a less positive picture of how that works in practise (Lee, Czaja, Sharit, 2009; Charness and Fox, 2010; Lesser and Rivera, 2006).

The benefits of employing older workers have led to pushes within a number of organisations to retain and re-employ seniors in flexible working roles. Home Depot for example offer 'snowbird specials' which consist of winter work in Florida coupled with summer work in Maine (Freudenheim, 2005).

This shift to the perceptions surrounding the value of older workers is occurring at a time when attitudes are changing towards financial responsibility. Governmental and private rhetoric across the industrialized west is in many cases consistent with the message that the future of retirement is not one in which social funding at a particular age can be taken for granted (see the introduction to this dissertation for some recent examples of changes). Even certain high schools are now incorporating retirement planning into their math curriculum (Ekerdt, 2004) demonstrating a change of attitude that reflects the deeper understanding we have achieved in the last few years with regards to the role of retirement and paid employment in the future.

However subtle biases and discriminations still exist, and legislation that was introduced to strengthen the protection of older workers has instead legitimized certain forms of discrimination (Sargeant, 2006). The older worker situation is still not ideal even if the evidence points towards improvements in the availability of employment and in self-proclaimed attitudes on the part of employers and managers.

2.3 Implications of an Aging Workforce

The shifting demographics have brought with them two specific problems that are unique to the aging workforce:

- Many industries cannot replace their retiring workforce.
- As individuals move into retirement, they take with them a lifetime of often irreplaceable experience and training

Additionally, the situation is further complicated by the psychomotor and cognitive profiles of older users – modern workforces are becoming increasingly computerized, and more employers are seeking employees who have computer literacy as part of the set of skills they bring with them. Even in cases where an older adult may have this computer literacy, the nature of the aging process brings with it sometimes subtle issues of psychomotor and cognitive decay (See Czaja and Lee, 2002s for an overview of these), and any sustainable attempt to incorporate older workers in such computerized environments will require, as a matter of course, a portfolio of cognitive and accessibility tools. These tools will be required to support older users with the often subtle interactions of minor ailments that act to hinder their ability to productively work with a computer system. It is this last point that is the focus of this dissertation.

In America, according to the Centre on Aging and Work, the demographics for workers aged 55 and over is likely to go from a 16.2% representation in 2005 to a 23% representation in 2050 (Pitt-Catsouphes and Smyer 2007). Some of the immediate problems may be mitigated by a generally positive pervasive attitude amongst older workers towards continued employment past retirement (Butrica, Schaner, and Zedlewski, 2006, Employers Forum on Age, 2007) - individuals who could retire are choosing not to for various personal, economic and cultural reasons (Pitt-Catsouphes, and Smyer, 2005), especially if organisations are willing to meet the changing employment needs and desires of individuals (Delong, 2006).

Many organisations are taking advantage of this pervasive attitude and bringing older workers back into the fold. Manpower Inc have developed a report discussing a number of ways in which this could be done, including providing 'per project' consultancy work for retired members. The core recommendations of the report suggest a framework of: changing retirement models, retention of existing employees, and a comprehensive knowledge transfer strategy (Manpower Inc, 2010). Part of this process is engendering a culture that honours experience (Streb, Voelpel, Leibold, 2008), and enhances individual roles as 'tribal elders'

2.4 Implications of a Retiring Workforce

We can already see the impact of retirement-related issues in various sectors, such as in the global chemical industry (DeLong, 2002a), public health (Hanson and Lesser, 2009), education (Encore, 2010), national defence (Liu and Fidel, 2007), the electrical industry (Lave, Ashworth, Gellings, 2007) and space travel (Delong, 2004). Petch (1998) has argued that we no longer have the capability to repeat our manned visit to the moon due to organisational knowledge loss. A refurbishment project for Trident missiles had to be put on hold for a year, at a cost of $69 million, due to informational attrition at the National Nuclear Security Administration (NNSA), as reported by the Government Accountability Office (2009):

> *In addition, NNSA had lost knowledge of how to manufacture the material because it had kept few records of the process when the material was made in the 1980s and almost all staff with expertise on production had retired or left the agency.*

It should be stressed too that this knowledge was lost when retirement was following a predictable trajectory. Recent studies have shown the majority of employers are not prepared for the mass retirements that will stem from the Baby Boomer generation exiting the workforce (Fisher, 2005), even if they do understand the risks (Shaw and Smith, 2003).

The consequences for a workforce to lose its most experienced members can be severe, and several industries are suffering very real problems because of the loss of knowledge, expertise, and contacts that are possessed by the older people within their workforce. Delong (2002b) identifies the three major risks that come with losing experienced members of staff:

- A reduced capacity to innovate
- The ability to pursue growth strategies is threatened
- Efficiency is reduced

Lesser (2007) states the problem this way:

> *When individuals leave an organisation, the organisation often loses a career's worth of experiential knowledge. This can range from e-mails, reports and documents that employees have accumulated, to tacit knowledge about how to do their jobs effectively. Too often, this knowledge leaves the organisation without any attempt to identify, capture and share it with others.*

> *As a result, remaining employees often search futilely for answers to questions that have already been answered, recreate analyses that have been conducted many times over or simply fail to heed previously learned lessons that were never formally identified and captured.*

Leonard and Swap (2005) talk about this as Deep Smarts; the intelligence that only comes from experience. They stress it takes at least ten years to build genuine expertise in a field, and that while those skills are often transferable, they cannot be built from scratch in a shorter time-frame. If it is only age and experience that allows for Deep Smarts to be built, it logically follows that there is a measurable impact on an organization when those who have the most age and experience retire without transferring some of their knowledge back to the organization.

It is important to note that these are not abstract, theoretical problems – several industries are suffering now, and more are going to suffer in the near future (See for example Hanson and Lesser (2009); Peterson and Spiker, 2005; Platman and Taylor, 2004). One especially well known example of the problem can be seen in how companies were required to out-source COBOL developers in the run-up to the Year 2000 (Paulson, 2001) – skills that had been deemed to be obsolete and expendable turned out to be critical to ensuring continued operational competency. The recent financial crisis is another case in point – a loss of financial experience has been highlighted as a causative factor, and part of the strategy to help resolve the crisis has been to rehire older, more experienced individuals who had previously retired (Aldrick, 2009).

However, knowledge management initiatives are fraught with difficulties. There are abundant examples of knowledge management tools succeeding (see Tapscott and Williams, 2006; Davenport and Prusak, 2000 for an overview of some of these), but there are also many examples of tools that were introduced but never developed the necessary traction to be successfully adopted. One of the canonical studies in this field is to be found in the work of Orlikowski (1992) – while this paper was written almost twenty years ago, its lessons are still relevant today. Two key elements are identified as leading to knowledge management failure:

- The mental model of individuals regarding the role of technology in their work
- The organizational properties with regards to social norms and reward systems

Knowledge management tools are not technological affairs – they are enshrined in complex social dynamics and it is not possible to extract them from the social context in which they operate (Giordano, 2007).

Additionally, knowledge sharing has a cost and must be offset with a benefit. If the cost is greater than the benefit, people simply will not contribute or consult. Finally, there is the need for a mechanism to allow for easy identification of relevance of information, and for reliability of information. People need to know how to find the information they need, and to be able to identify how useful that information is.

There are several features of older workers in particular that suggest that knowledge management might succeed better with this demographic group. It plays into the recognized psychological period known as ‘summing up' (Cohen, 2007) in which individuals wish to give back some of the knowledge and experience they have accumulated over their life-time. Additionally, it also supports the documented change of preference towards knowledge-based roles (Hanson and Lesser, 2009). Finally, an effective knowledge management tool allows for recognition of expertise, and for the emergence of individuals as ‘tribal elders’ within their own social networks.

Knowledge management in organizational contexts is not a solved problem, and the additional considerations that are raised by applying technical solutions to the problem within older demographic groups are not well served in the literature. While it is not a focus of this dissertation, this area was explored in a small pilot study conducted during this research, where social psychological tools were employed in an attempt to generate sustainable contributions to a collaborative knowledge base. Thus, while the issues associated with a retiring workforce are not specifically germane to the research to follow, it represents a second major area of investigation for those concerned with the the issues associated with broad demographic changes.

2.5 Older Users in the Workplace

Increasing the age of retirement has numerous implications for the workforce. It requires training strategies that emphasize involvement and opportunities over the entire course of an individual's working life. One of the complaints that older workers have is that they are denied access to the same training opportunities as their younger colleagues (Hanson and Lesser, 2009). Additionally, the training preferences of older workers often do not match those that are used in training situations - individual and small group work are more effective with older individuals than lecture and classroom scenarios (Dickinson, Eisma, Gregor, Syme and Milne, 2005; Hawthorn, 2005; Lesser and Rivera, 2006), and peer support is particularly important for those lacking in confidence (Alm, Gregor and Newell, 2002). The increased cost that is associated with these training options can make them less attractive to an organisation seeking to ensure that the benefit they receive from training is commensurate with the cost (Pilz, 2009).

Coupled to this is a persistent impression amongst management and the population at large that older workers are not interested in training opportunities and have less desire to learn new and necessary technology (Notess and Lorenzen-Huber, 2007; Czaja, 2005) - in many organisations, older users simply do not get the support needed to remain valuable employees with up-to-date skill-sets (Hanson and Lesser, 2009). Providing strategies for counteracting this is complicated by the fact that most managers are attitudinally positive about older workers (McMullin and Tomchick, 2004; Peterson and Spiker, 2005). The attitude is positive, but the trends in provision of training opportunities and studies conducted with older workers as participants tell a different story.

While this assumption with regards to the attitudes of older users is not well supported by modern literature (Alm et al, 2002; Convertino, Farooq, Rosson, Carroll and Meyer, 2007; Lesser and Rivera, 2006), older workers do come with an increased range of cognitive and accessibility issues that interfere with their ability to make use of modern technology, as well as more traditional machinery. Aging carries with it general across the board declines in visual, auditory, psychomotor (Czaja and Lee, 2002) and working memory capabilities (Sharit, Hernandez, Czaja and Pirolli 2008). However, research has also shown that older users can compensate for a decline in cognitive ability through the increased use of crystallized intelligence (Nair, Czaja, and Sharit, 2007). Where an information task is ill-defined, the performance of older users in information technology tasks can actually exceed that of younger users (Chin, Fu, Kannampallil, 2009). All of these issues have implications for older users and technology. The end result though is that older users are not only willing to make use of new technology, they are capable of doing so given appropriate social and technological support (For examples, see: Zajicek, 2003; Hanson, 2002)

2.6 Cognitive and Accessibility Support for Older Users

Most modern operating systems come with accessibility support built-in. Computer-assisted technologies have the potential for improving the user-experience of people with cognitive, visual and psychomotor issues. However, older users cannot be easily categorised in terms of their need for accessibility or cognitive support. Older users generally do not have one specific issue that needs to be addressed, but a unique combination of issues. There is rarely one obviously serious ailment, but instead a subtle blend of minor ailments, none of which may be considered substantial enough in itself to be self-reported as a disability (Gregor , Newell and Zajicek, 2002). Additionally, the amount of variation between individuals tends to increase as age increases (Gregor et al, 2002). Thus no one specific assistive technology element is likely to suffice, and even if it were it is not assured that an older user would believe they have a need for it. When we talk about the accessibility concerns of ‘older users’, we are actually referring to a multitude of combinations of ailments, something Gregor et al (2002) refer to as ‘dynamic diversity’.

The impact of this dynamic diversity is that even perfectly effective accessibility solutions aimed at supporting a single disability may be ineffective when faced with an older user. Pullin and Newell (2007) give an example of this:

> *Thus, for example, speech synthesis, which is a very successful accessibility option for people with visual impairment, will not work for many older people because of reduced hearing coupled with the high cognitive load needed to cope with the reduced comprehensibility of synthetic compared with natural speech.*

The level of computer familiarity amongst older users tends to be lower than than of younger adults (Fox, 2004). Even the accessibility tools that are built into most operating systems require a disproportionate amount of computer knowledge to find and configure (Trewin, 2000). Many users would be adequately served by the provided operating system facilities themselves, but these require the user to know the tools exist, how to switch them on, and what the implications are of the changes they are making. Hawthorn (2003) has this to say:

> *As an aside I tend to rule out solving older users' problems by way of "accessibility features" such as adjusting the system fonts or mouse response characteristics, at least in MS Windows. The older users I work with have only a limited chance of successfully finding out about such features, understanding their impact, locating them and making useful adjustments. Once such adjustments are made they can be hard to turn off yet they can negatively affect the functionality of other programs that have not been designed to consider the possibility of accessibility features being turned on. Further many of the supporters of older users that I have worked with do not, themselves, understand the range of accessibility features available.*

When accessibility support is provided, logistical problems are encountered. In a traditional office environment where one individual uses a single computer, these do not arise. However, many modern workplaces are moving across to hot-desking and shared computing resources. It takes time to properly configure a piece of software, and accessibility software, such as voice recognition software, often requires substantial investment to train.

Finally, there is an element of social stigmatisation that comes with the use of accessibility software and technology designed to be used by the elderly (Sokoler and Svensson, 2007). Older users may actively choose not to use the tools they have available because they perceive that their use reduces their status in the eyes of their peers (Sokoler and Svensson, 2007). They do not want to feel as if they need extra support to do the job others can do without help and so they willingly choose not to be supported.

Good accessibility and cognitive support for older users then has the following features:

- Non-stigmatizing
- Easily usable by computer novices
- Can help cater for dynamic diversity

- Supports mobility between computer systems

A later chapter of this dissertation will discuss a tool designed to provide these features, and some of the plug-ins that have been developed to show its utility and value.

2.7 Conclusion

The aging population brings with it new challenges, but also new opportunities to maximize the benefits older workers can provide in the workplace. To do this effectively, we need to first understand the specific issues that affect older workers. There is tremendous risk associated with older workers leaving the workplace – organizational knowledge is a genuine business asset that gets eroded every time an experienced member of staff retires. An organization positioning itself for maximum benefit in an aging society will have processes in place to stem this information attrition and allow those who have accumulated decades of experience to pass that experience on to their younger colleagues. However, it is perhaps more important still to put in place support for those workers who wish to continue their employment beyond the traditional age of retirement.

In 1952, the Queen sent out less than 3,000 congratulatory messages to new centenarians. In 2005, that number rose to 6,914 with an additional 24,304 messages for diamond wedding anniversaries (Monarchy Today, 2005). Older people in society have an incalculable cultural value that comes from what Gene Cohen has called Developmental Intelligence – the intelligence that comes from wisdom, rather than from purely intellect. The research shows that organisations have an opportunity to gain tremendous value from this by incorporating older people into their workforces and for ensuring that their existing workforce of older individuals can continue to contribute in an increasingly computerized environment.

The UN summarise the demographic situation as follows (United Nations, 2002):

- *Population ageing is unprecedented, without parallel in human history - and the twenty-first century will witness even more rapid ageing than did the century just past.*
- *Population ageing is pervasive, a global phenomenon affecting every man, woman and child - but countries are at very different stages of the process, and the pace of change differs greatly. Countries that started the process later will have less time to adjust.*
- *Population ageing is enduring: we will not return to the young populations that our ancestors knew.*
- *Population ageing has profound implications for many facets of human life.*

The greying of the population has been characterized as an 'aging crisis' in the popular press(such as in McNicoll, 2009; Peterson, 1999; Chang, 2010; Bryskine, 2011), but Scotland's Futures Forum

(2007) takes the more realistic view that it is not a crisis, but is instead a challenge – and like any challenge it has its associated payoffs and its associated costs. There is a danger should it go unaddressed, but there is much to feel positive about with regards to capturing the capabilities of older people in the workplace.

Incorporating older workers effectively also means changing the way businesses treat those approaching retirement age. When training is tailored to the specific requirements of older workers, it is far more effective than it would be otherwise. Older workers are often disadvantaged in terms of provision of training – an effective strategy for making use of older workers is to ensure that training is provided equitably across all age groups.

Finally, an organization should make an effort to properly support the accessibility and cognitive needs of its workers, and to do so in a way that is not stigmatizing. Aging brings with it a unique portfolio of accessibility and cognitive declines, and any solution provided should be cognizant of the fact that there is no such thing as an 'old person' as far as accessibility is concerned. Every older person benefits differently from accessibility support, and a unique tailored solution may be required for each

3 Current Accessibility and Cognitive Support

3.1 Introduction

As the workforce and society in general become increasingly computerised, more and more employers are calling for individuals with computer literacy (Leahy and Dolan, 2009). In the previous chapter this dissertation discussed the implications of an aging workforce, and the need for organisations to incorporate older workers. The dissertation also briefly discussed those financial pressures on older individuals, and society as a whole, that are changing our current conception of 'retirement'. However, when discussing computer literacy and older adults, the issue is complicated by a number of factors – some of these are psychological, and others are a result of the natural psychomotor, perceptual and cognitive declines associated with aging.

Much of the research in the field focuses on software tools to correct or compensate for the physical issues. Such research falls into a broad category of tools known as 'accessibility' – research designed to remove hurdles from older users when they are interacting with software systems.

A second broad class of software support may be thought of as 'cognitive support' (a useful overview of tools and techniques of this nature may be found in Hollender, Hofmann, Deneke and Schmitz, (2010). These tools are designed to lower the cognitive burden for users, either by simplifying the workflow required to accomplish common computing tasks, simplifying the language (Newell, Carmichael, Gregor, and Alm, 2003), or by providing software tools that permit some of the cognitive burden of working with a computer to be shared with the computer itself (such as discussed in Lindqvist and Borell, 2010).

The third broad class of research is primarily educational, and focuses on the specific training needs of older users in an attempt to make existing systems more comprehensible without altering the specifics of interaction.

This chapter will discuss these three broad categories in relation to both the psychological and physical concerns of older adults.

3.2 Defining the Older Adult

The literature is not consistent on the convention adopted when discussing what is meant by an older adult. Various informal vocabularies have emerged within different subgroups of researchers, but the specific boundaries between age groups, and the number and naming of these age groups themselves, are inconsistent (Convertino et al, 2005), and individuals identified as 'older worker' in the literature may range between the ages of forty and seventy five. Within this dissertation, the phrase 'older worker' will be used to identify individuals of age sixty and over, to harmonise with the findings in the literature regarding the rate and timing of physical and cognitive decline. The term 'older user' will be used synonymously.

The issue of naming conventions would be one of limited importance in this discussion if it were not the case that individuals vary considerably even within a relatively narrow age band. While in this dissertation, the term 'older user' will be used as if to identify a specific demographic profile, it is important to realise that there are very substantial differences between a fit person at the youngest point in the definition, and the frailest at the farthest point. There is more variability within the 60+ age demographic than in any other age group. A substantial percentage of individuals who are over 65 have an impairment serious enough to impact on quality of life (Pullin and Newell, 2007), but equally there are many eighty year olds who have no substantial self-reported disability – there is no such thing as a 'representative' older user (Newell and Gregor, 2001) who can act as a proxy for an entire demographic group.

Likewise, individual older workers cannot be easily categorised in terms of their need for assistive technology. Rather than having one specific issue that can be addressed in isolation, older people instead tend to have a subtle blend of minor ailments, none of which may be considered serious enough to be self-reported as a disability (Gregor et al, 2002). Thus, no one specific assistive technology element is likely to be sufficient to meet the accessibility needs of a 'fit' older person while also meeting the needs of a frail individual at the higher end of the age-range. As the level of variation between individuals tends to increase as age increases (Gregor et al, 2002) - solutions that may have been effective at age 60 can become less so at 70, as in the Pullin and Newell (2007) example of speech synthesis given in the previous chapter.

Such complex interactions of minor ailments frustrate attempts to define from the top-down any kind of consistent and universally applicable framework for designing interaction technologies for older adults.

3.3 Software Development and the Older User

Software is generally not designed for older users (Czaja and Lee, 2002). Instead, it is designed for a group of people much like the individuals who develop it – relatively young, technologically savvy developers. The typical software developer finds it more difficult to develop software for groups with a substantially different profile of wants, needs and priorities (Keates and Clarkson, 2002).

The tendency for older users to blame themselves rather than poorly designed computer systems (Newell, Dickinson, Smith and Gregor, 2006; Newell, Arnott, Carmichael, Morgan, 2007) combined with the fact most computer developers have little direct experience of working with older users (Newell et al, 2006) and negative software developer attitudes (Knight and Jefsioutine, 2002) creates a situation whereby even identifying the flaws in the software development process with regards to designing software for older users is not at all straightforward. Older users tend to be very positive about software prototypes (Newell et al, 2007) even when the desire is for an honest, critical evaluation.

Part of the problem lies though not with the specific issues of older users, but instead in the culture of software development. While there are abundant examples of guidelines and principles that have been designed with developers in mind, these guidelines are often lacking in context and when applied can sometimes do more harm than good (Thatcher, 2003) because they must summarise often very complicated and nuanced guidance into a relatively small, comprehensible set of instructions (Milne, Eisma, Gregor, Sloan, Dickinson and Carmichael, 2005; Sloan, Heath, Hamilton, Kelly, Petrie, Phipps, 2006).

Partially the problem in software development is that popular development processes such as User Centred Design have no guidance to offer when developing software for older users (Newell, 1993; Newell and Gregor, 1997) because that guidance is located in the more rarefied academic papers of conferences and journals.

Even for those who wish to develop software in line with the accessibility needs of older users, guidelines do not form a comprehensive theory (as discussed in Scheiderman's foreword to the Department of Health Usability Guidelines) in a form that is both tractable and respectful of the context (Thovtrup and Nielsen, 1991). This latter point is important, for understanding the context of guidance is one of the key elements in applying it correctly (Milne et al, 2005) – otherwise, such guidance can seem overly cautious and not genuinely representative of real user interactions (Knight and Jefsioutine, 2002). Descriptions of user difficulties with computer systems likewise can seem exaggerated (Newell et al, 2006) with developers often feeling that these outlines provided by academics must be the rarest, worst-case, most extreme examples.

Even software developers working actively with older users must often see for themselves the difficulties older users have with computer systems before accepting the conclusions of researchers on the topic. Controlled exposure of software developers to older users can greatly increase the ‘buy-in’ needed to progress software development along accessible routes. Such direct involvement has several beneficial knock-on effects, as older users can often be the trigger for a kind of ‘mutual inspiration’ between developer and user (Eisma, Dickinson, Goodman, Mival, Syme and Tiwari, 2003) whereby the developer inspires the user to answer questions that are not being asked, and the user inspires the developer to create genuinely innovative solutions that are actually useful for older users.

It is not enough to simply be mindful that novice users may be interacting with a system. Many of the issues that older users have with such technology are such an ingrained part of the mental toolkit of a software developer that they simply never even occur as a possible point of interface complication. For example, the concept of 'double clicking' (Newell et al, 2006), dragging windows (Hawthorn, 2005) or even the words 'select' (Hawthorn, 2005), 'menu' and 'toolbar' are all concepts which have caused problems for older users approaching a new and unfamiliar computer system. Such language forms a largely subconscious part of a software developer's vocabulary, and even guidance to 'keep it simple, stupid' can fail to help a developer simplify their choice of language and jargon because they simply do not realise that the words they are using are not universally understood. Additionally, inconsistency of jargon between different interfaces, and even within the same interface, can result in multiple sets of terms relating to the same interaction choices

Common metaphors of interaction that are suitable for experienced users can likewise cause problems for novices – issues such as double clicking (Dickinson et al, 2005), highlighting text fields (Ellis and Kurniawan, 2000), and using the scrollbar (Nielsen, 2005) have all been reported in the literature. Even the almost ubiquitous concept of 'saving' is one that has few parallels in other domains (Dickinson et al, 2005).

A more general case of inconsistency comes from different versions of the same application, or indeed the same version of an application on a different platform. Applications such as Microsoft word can be tailored to specific, individual preferences – but deviation from the norm can frustrate those users who rely on routine to interact with a system (Dickinson et al, 2005).

Studies conducted during this research also revealed other such problems relating to the hardware itself, including differing tactile feedback on keyboards, and sensitivity of mouse and touch-pads. These issues may not be substantial in and of themselves, but they contribute to a general inconsistency in computer interactions.

3.4 Experiential Issues

In psychological terms, older people are less likely than younger people to have used computers before, and may feel anxious about using them (Chadwick-Dias, McNulty, Tullis, 2003), a result which is demonstrated in other earlier studies such as Ellis and Allaire (1999) and Czaja and Lee (2001). Older adults typically self-report lower levels of confidence with computers than younger counterparts (Marquie, Jourdan-Boddaert and Huet, 2002). Lack of previous exposure makes it more likely that older users will have initial negative experience with computers which may dissuade from future use (Dickinson et all, 2005). Usability flaws in a system can have substantial negative impact on the willingness of older users to continue with a system (Newell et al, 2007), and indeed even for experienced users, user interface flaws can be frustrating (Ceaparu, Lazar, Bessiere, Robinson, and Schneiderman, 2004).

The level of computer familiarity amongst older users tends to be lower than that of younger adults (Fox, 2004) and even the available accessibility tools built into most operating systems require a disproportionate amount of computer knowledge to find and configure (Trewin, 2000). People with disabilities, especially older adults, are less likely to use computers in general (Fox, 2004).

Coupled to this is a powerful and common sentiment that computers simply aren't useful or worthwhile for older users. In a newspaper column for the Independent, Joan Bakewell (2008) writes:

> *The other option as you get older is to ignore the latest fad and opt for the moment in the technological evolution where you felt most comfortable. Plenty of writers still write longhand. I know a number of my generation no longer drive, who don't have a mobile, who never learnt to handle emails and who appear to live lives of serene contentment. I'm sometimes inclined to envy them.*

Outside of an employment context, the perceptions of older users can complicate attempts to introduce technology – either as a result of their not seeing a use for it in daily life (Morrell, Mayhorn and Bennett, 2000), or as a more active withdrawal from interaction as evidenced by those who self identify as 'non users' (Wyatt, 2003). The situation is complex and other issues older users have identified include lack of interest, the speed of technological progress, and the cost of devices (Morris, Goodman and Brading, 2007), although work by Mitzner, Boron, Fausset, Adams, Charness, Czaja, Dijkstra, Fisk, Rogers and Sharit (2010) shows that on the whole attitudes are more positive than negative. Willingness to adopt computer use is highly dependent on application, with older users showing a desire to learn specific technology such as word processors, email, and the world wide web (Goodman, Syme and Eisma, 2003; Hanson, 2001). Willingness to learn technology may perhaps be able to be taken as a given when discussed in the context of older workers, but the underlying attitudes remain important.

3.5 Impact of Aging

One of the consequences of the aging process is a decline in sensory acuity. The literature on this topic is abundant, and in this section I shall discuss some of the better known examples of the work. However, it is important to note that while many of the studies cited are highly referenced in the field, older adults are becoming healthier and better educated with every demographic study (Czaja, 2005). This has implications, outside the scope of this dissertation, for the conclusions drawn in older respected publications.

3.5.1 Visual Decline

In terms of visual acuity, there is evidence to suggest that eye problems tend to appear in the early forties (Fozzard, 1990). At this stage of development, the ability to discern fine details is reduced, and individuals may notice difficulties in adjusting visual focus between the near and far as well as problems with visual acuity generally (Schieber, 2006). Additionally, detecting differences in contrast may become difficult (Owsley, Sekuler, and Siemsen, 1983; Schieber 2006), and it also becomes more difficult to pick up differences in colour (Helve and Krause, 1972; Schieber, 2006). Older users may also demonstrate a reduction in the width of their visual field (Cerella, 1985), and the ability to accommodate to dim lighting conditions (Schieber, 2006), as well as a lowered ability to discern temporally contiguous events (Schieber, 2006). Kline and Scialfa (1996) provide further work on the subject of decline of visual performance indicators.

Coupled to this is an increasing reduction in the speed regarding processing of visual information (Kline and Szafran, 1975) – this causes additional difficulties in detecting figures within figures (Capitani, Della, Lucchelli, Soave, Spinnler, 1988), an inability to recognise fragmented or incomplete objects (Salthouse and Prill, 1988; Frazier and Hoyer, 1992), and relative slowness when performing location tasks (Plude and Hoyer, 1981). However, it must be stressed again that any specific older individual, especially those in the category of the ‘oldest old’, can demonstrate very marked differentiation between the fittest and the frailest members, and that for some of these impairments, eyeglasses can serve as an effective compensation up until the age of 88 (Schieber, 2006).

3.5.2 Auditory Decline

As with vision, hearing sensitivity too declines with age, with around a fifth of adults aged between 45 and 54 having some hearing impairment, with that figure rising to 75% for those between 75 and 79 (Fozard, 1990; Kline and Scialfa, 1996). Additional difficulties include missing ‘attention grabbing’ sounds, even those in safety critical devices such as smoke alarms (Berkowitz and Casali, 1990; Huey, Buckley, Lerner, 1994). The inability to discern high-pitched tones can result in parts of speech being missed and the older adult having to guess at meaning, with the oldest old perhaps missing as much as 25% of a conversation (Feldman and Reger, 1967). Speech generally is an issue highlighted with regards to older people – age brings with it greater impairment on speech perception tasks such as identifying speech in noise, with a difficulty in discerning between voiced and voiceless consonants being particularly notable (Thornton and Light, 2006).

While auditory prompts are not as critical to most software applications as visual cues, these issues are relevant to topics such as speech synthesis, voice recognition, and attention-getting auditory warnings.

3.5.3 Psychomotor Decline

Age brings with it numerous impairments in the sphere of psychomotor activities. Among these impairments are a reduction in response times on complex motor tasks (Spiriduso, 1995). When there is incompatibility between stimulus and response (such as in Stroop, 1935), the age of the individual under examination has an impact on the lengthening of response times.

To go along with decline in response times, older users also demonstrate poorer performance than younger users when asked to track a target (Jagacinksi, Liao, Fayyad, 1995), and are slower when capturing a target with a mouse (Walker, Philbin, Fisk, 1997). This effect is especially marked when the target is small, such as words on the screen (Charness and Bosman, 1990). Speed of movement is also an issue in such tests, with the largest difficulty in acquiring a target being linked to the distance and the size of the target (Chaparro, Bohan, Fernandez, Choi, Kattel, 1999).

Some of these impairments can be compensated for with experience and recent practise (Krampe and Ericson, 1996), such as when individuals develop a strategy for the movement they need to make with regards to acquiring mouse targets (Walker, et al, 1997). Many of the studies done on these topics with regards to computer use are performed under laboratory conditions and there is evidence to suggest that the decline may not be so marked in regular, non laboratory tasks (Salthouse, 1984) when older users can plan in advance for tasks to come (Salthouse, 1984; Bosman, 1996). Czaja and Sharit (2003) note the problems associated with laboratory tests and their relevance to the ability of older adults to perform in real life. Charness and Holley (2001) have noted that target selection difficulties can be alleviated in part by removing the layer of abstraction represented by the mouse and replacing it with an alternate input device such as a light-pen or a stylus.

3.5.4 Cognitive Decline

While it is true that older adults demonstrate cognitive difficulties in making use of aspects of modern technology (Sharit, Hernandez, Czaja and Pirolli, 2008) it has also been demonstrated that appropriate training aimed at those areas in which cognitive decline are most marked can improve performance (Sharit et al, 2008). However, in saying this it is important to realize that cognitive decline is, in the main, unavoidable and not limited to those traditionally regarded as 'older adults'. Nielsen (2008) identifies a trend in people between ages 25 and 60 - for every year of age, time-related performance with regards to web tasks declines by 0.8%. We are all getting slower, all of the time. We can all benefit from the reduction of the cognitive load required in using modern technology, especially information seeking (Sharit et al, 2008). Cognitive support thus should not be seen as a domain purely for the elderly, but it is this age group where the impact is more marked.

In selective attention tasks, the ability to ignore distracting information declines with age (Connelly and Hasher, 1993; Kotary and Hoyer, 1995). In divided attention tasks, which are common to many computer interactions, the situation is more nuanced – declines in performance associated with age have been noted (such as in McDownd and Craik, 1988), but such declines seem associated with complex tasks only. Some of these issues may not come down to the issue of attention specifically, but instead may be a response to a more general reduction in cognitive resources available for performing such tasks (such as is argued in Korteling, 1994; McDowd and Craik, 1988).

Modern user interfaces in particular are becoming increasingly cluttered, and there is experimental evidence to suggest that complicated graphical displays can deter natural exploration and experimentation with an interface as far as older users are concerned (Wright, Belt and John, 2003). It is those with higher cognitive resources that are most likely to adopt new technology (Czaja et al, 2006), and perhaps some of the low take up of technology is linked to the issues outlined above.

3.5.5 Memory and Intelligence

Working memory is used as a temporary store for short-term data, and plays a role in the manipulation of such information in complex cognitive tasks. Some researchers (for example, Zajicek, 2003) refer to related measures of fluid memory and crystallized memory – the cognitive literature more properly relates to these as fluid and crystallized intelligence but they have relevance to the topic of memory. Fluid intelligence is closely related to working memory, and represents the ability of an individual to perform tasks such as problem solving, reasoning and pattern recognition. Crystallized intelligence is the ability to use skills, knowledge, and previous experience. It is not a measure of memory in itself, but it does rely on long term memory storage in an individual.

Working memory is an issue for older adults, and working memory is tied to performance in a range of tasks. It has been linked to performance in search (Czaja et al, 2001), and web navigation (Laberge and Scialfa, 2005).

However, the impact of the aging process on memory is nuanced, with several studies identifying particular categories of memory that are not impacted. Crystallized intelligence is relatively unaffected by age whereas fluid intelligence in particular is more likely to suffer as a result of cognitive decline - the findings reported by Backman, Small, Wahlin and Larsson (2000) can be summed up as ‘everything declines, but crystallized intelligence declines slower’. Younger adults will, almost by definition, have less experience to call upon than older adults, and relevant accumulated knowledge can counteract a decline in fluid intelligence (Rabbitt, 1993).

Memory too has more subtle impacts on task performance, with many studies showing that domain specific knowledge contributes to success in performing cognitively demanding related tasks such as in playing chess (Chase and Simon, 1973), bridge (Engle and Bukstel, 1978), and baseball recall tasks (Hambrick and Engle, 2002). Domain knowledge in these studies was shown to have a much stronger impact on performance than working memory and the age of participants. Related to this, prior experience with computers has been shown to be an important predictor of performance (Czaja and Sharit, 1998; Czaja et al, 2001; Charness, Kelley, Bosman and Mottram, 2001).

3.6 Current Accessibility Research

There are many examples of accessibility and cognitive support tools that have been developed as research prototypes. Notable examples of these include SteadyClicks (Trewin, Keates and Moffat, 2006), the Invisible Keyguard (Trewin, 2002), the Dynamic Keyboard (Trewin, 2004) and hardware solutions such as external mouse drivers (Levine and Schappert 2005). Dickinson, Eisma and Gregor (2002) outline two further specific tools (Piloot and SeeWord) designed to ease cognitive burdens, although these tools are not aimed specifically at older users.

Additional work has been done on the concept of 'training wheels' interfaces, where advanced functionality is incorporated incrementally (such as outlined in Carroll and Carrithers, 1984), and Schneiderman (2002) stresses the value of such interfaces and variations such as minimal manuals (Black, Carroll, McGuigan 1987), task-centred design (Vassileva, 1996) and plastic user interfaces (Thevenin and Coutaz, 1999). Other work on this subject can be found in Czaja and Lee (2001), Newell et al (2002), and Hawthorn (2003).

Accessibility support software tends to focus on areas of the computer system where low level system input and output (I/O) occurs. This has two main impacts – the first is that the barrier to participation of researchers is greatly increased since modern virtualized programming environments have successively abstracted away from the system layer, making it more difficult for developers to trap and manipulate system-level I/O events. The researcher requires a high degree of programming knowledge to be able to achieve what are often very modest goals.

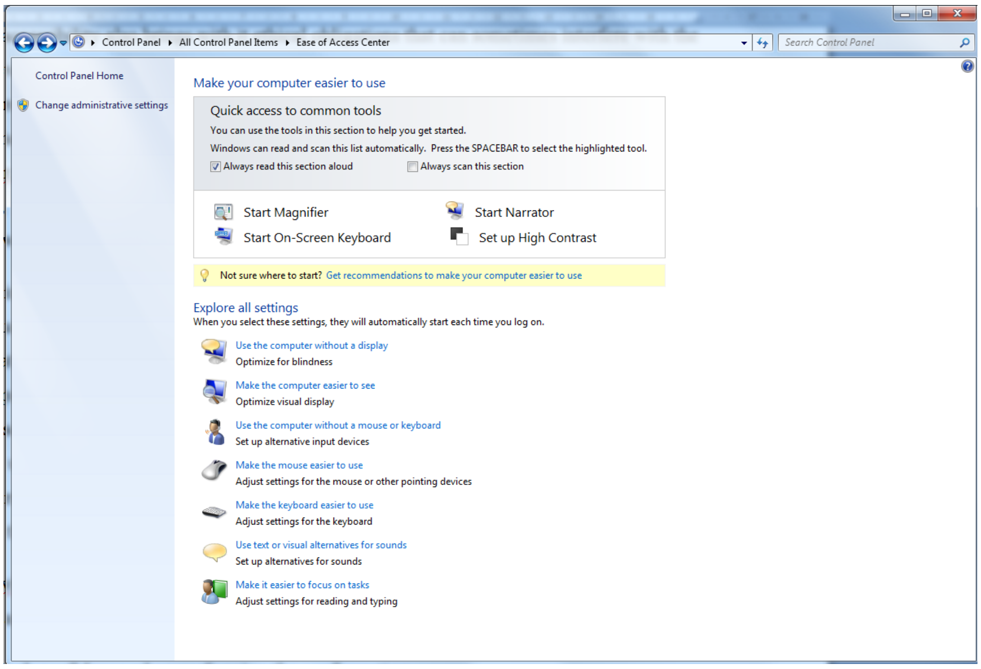
In order to counteract this, the research software has to be tightly bound to a particular operating context – software is usually tested and deployed on Windows machines, with no guarantee or expectation that it would work similarly or at all on other platforms (as noted by Hanson et al, 2005). This perhaps explains the large focus of the accessibility research community on issues of the web where platform concerns are less pressing, such as in Kelly, Sloan, Brown, Seale, Petrie, Lauke and Ball (2007); Richards and Hanson (2004); Zaphiris, Ghiawadwala and Mughal (2005); Hart and Chaparro (2004); Zaphiris and Sarwar (2006); and Zajicek (2007) amongst many others. The modern focus is primarily on the web, and this is a shift away from the interaction issues associated with the desktop.

Related to this, a relatively large project called Raising The Floor (http://raisingthefloor.net/) is seeking to draw together individuals across organisations and fields to share research and collaborate on increasing the accessibility of web and mobile technologies. Vanderheiden (2008) has talked about the need for a new focus on micro-assistive technology, though the RTF focus on web technology has moved the initiative into areas to which the tool discussed in this dissertation is not well-suited to contribute.

Research into accessibility also brings with it ethical obligations that can sometimes interfere with the reliability of evaluative feedback (Alm, 1994). Also there is no guarantee that even successful trial runs of software will result in an operational product being developed or sold. Much research has been done at the academic level on software tailored for aging users, such as email clients (Dickinson, Gregor, Mciver, and Milne, 2005), web-browsers (Dickinson, Newell, Smith and Hill, 2005; Hanson, 2001; Hanson, 2002), and web portals (Newell et al, 2006) – but few of these innovations are available on a wider-scale for a more general audience, with the Cybrarian tool discussed in Newell et al (2006) being a notable exception[1].

Modern operating systems have numerous accessibility tools built into them, and these tools are often of very high quality. Windows 7 for example comes complete with settings for altering mouse speed, keyboard repeat and debounce rates, magnifiers for the partially sighted, and tools to allow those with movement impairments to use the keyboard rather than the mouse for cursor movement. It also comes with a screen reader and an on-screen keyboard for those who cannot use a physical device (such as those using a mouth-operated mouse). The range of tools offered support a wide range of physical impairments, and by virtue of being part of the operating system themselves they are tightly integrated into the user experience.

[1] The MyGuide service (http://www.myguide.gov.uk/myguide/MyguideHome.do) was produced from Cybrarian, and had an estimated reach of 500,000 people. This service is unfortunately scheduled to close in October 2011.



Control Panel ▸ All Control Panel Items ▸ Ease of Access Center

Search Control Panel

Control Panel Home

Change administrative settings

Make your computer easier to use

Quick access to common tools

You can use the tools in this section to help you get started.

Windows can read and scan this list automatically. Press the SPACEBAR to select the highlighted tool.

☑ Always read this section aloud ☐ Always scan this section

Start Magnifier

Start Narrator

Start On-Screen Keyboard

Set up High Contrast

Not sure where to start? Get recommendations to make your computer easier to use

Explore all settings

When you select these settings, they will automatically start each time you log on.

Use the computer without a display
Optimize for blindness

Make the computer easier to see
Optimize visual display

Use the computer without a mouse or keyboard
Set up alternative input devices

Make the mouse easier to use
Adjust settings for the mouse or other pointing devices

Make the keyboard easier to use
Adjust settings for the keyboard

Use text or visual alternatives for sounds
Set up alternatives for sounds

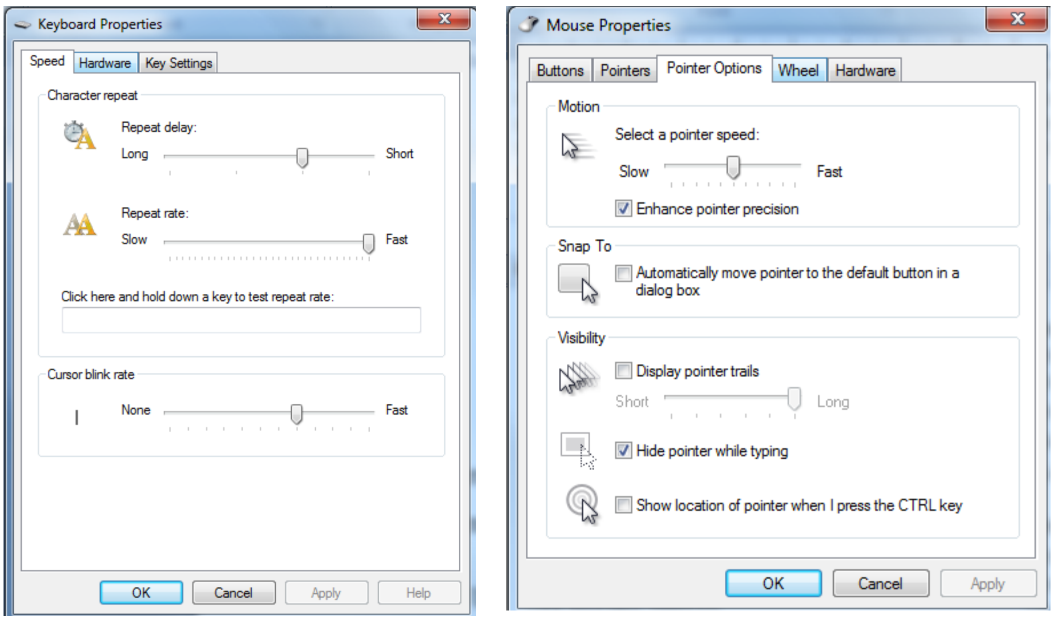
Make it easier to focus on tasks
Adjust settings for reading and typing

Figure 1 – Ease of Access Control Panel in Windows 7

However, Trewin (2000) notes the following substantial issues with current provision of accessibility, it is often the case that:

- Users lack confidence when performing the configuration
- Users lack knowledge of how to change the configuration.
- Users lack awareness of the options that are available.
    - Trewin (2000) cites an earlier study in which users with accessibility needs had limited formal instruction, and had to rely on experimentation or help from friends, colleagues and family.
- Users find it difficult to identify what the cause of a problem may be.
    - An example of this provided by Trewin (2000) relates to the difficulty in determining the correct course of action when encountering problems related to typing repeated characters.
- When working with an un-configured interface, users are sometimes operating under restricted circumstances that frustrate their ability to change settings.
    - As an example of this, users with physical impairments may find it difficult, or even impossible, to access the control panel to make the changes needed.

The complexity of an average control panel in most operating systems is problematic for users who are unfamiliar of the impact of changes, and the jargon used. Two examples of such control panels are shown in figure 2:



Character repeat

Repeat delay: Long — Short

Repeat rate: Slow — Fast

Click here and hold down a key to test repeat rate:

Cursor blink rate: None — Fast

Motion

Select a pointer speed: Slow — Fast

[✓] Enhance pointer precision

Snap To

[ ] Automatically move pointer to the default button in a dialog box

Visibility

[ ] Display pointer trails

Short — Long

[✓] Hide pointer while typing

[ ] Show location of pointer when I press the CTRL key

Figure 2 – Sample Control Panels in Windows 7

While the control panels shown above would be perfectly understandable to a computing expert, they are full of jargon that is confusing for novices – ‘repeat delay’ and ‘repeat rate’ are subtle terms with behaviour that may be non-obvious even when they can be tested. ‘Pointer precision’ is a setting that even computing experts may not know the implications of. Even the use of standard computing terms such as ‘dialog’, ‘default’ and ‘pointer’ was raised as an issue during studies conducted as part of this research.

The result is an environment for configuration that puts at a disadvantage those users who would most benefit from the accessibility support. The issues raised by Trewin (2000) in particular are of great importance to the accessibility framework described later in this dissertation.

3.7 Conclusion

Computers are complex, logically malleable devices encased in a culture of jargon. The current crop of ‘young computer users’ have grown up with computers being ubiquitous devices, and as a result have formed mental models regarding when and where particular kinds of interactions are appropriate. Older users, coming to computers for the first time later in life, do not have these mental models to fall back on.

Coupled to the inexperience typical of older adults and computers is a suite of ailments brought upon by the natural decline associated with the aging process. The default computer configuration requires relatively fine grained performance in a number of areas, such as motor control (for interacting with a mouse and keyboard), working memory and fluid intelligence (for processing the complexity of computer interfaces), and visual acuity (such as acquiring targets in a complex interface). Minor impairments in any of these areas can have complex effects due to the interrelationship of subtle ailments, even when an individual may self-report themselves as having no substantial accessibility concerns.

The difficulties associated with using the accessibility tools built into modern operating systems speak to the need to consider an alternate approach to providing accessibility support for users – if users cannot find the support, then perhaps the support should find the users.

Finally, the relative breadth and depth of accessibility research highlights an additional concern for those involved in the field – the relative complexity of developing software to support accessibility needs and having it work appropriately across platforms and contexts. Software developed for Linux is often markedly different from software developed for Windows, and the more tightly the software must interact with user input streams, the more different it becomes. Getting access to keyboard input is a simple task on Windows, requiring less than ten lines of C code, and similarly for getting mouse input. Windows as part of its standard API gives functions for letting developers hook into the standard event loop. However, the code that does this on Windows is entirely different from the code that does the same thing on Linux requires polling of the keyboard status port and an architecture for managing that without negatively impacting on system performance. The two approaches are entirely different, and so the code to do this must be written separately for each system. Neither of those approaches will work correctly on Mac OS X, and so different code still must be written to handle this. Windows makes use of a database to store application settings, while Linux tends to use flat text files in particular directories (and the directories in question will vary depending on the specific distribution being used). Simple programs that never touch on these issues can be easily ported from system to system, but more complex applications require the developer to understand the nature of the deployment platform and write multiple versions of the same program for each destination system.

The disproportionate investment in development time to produce even modest outcomes suggests that alongside the need to provide a more effective way for users to configure their interfaces, there is also a need to lower the developer burden required to provide accessibility support for users.

Chapters five and six of this dissertation discuss the ACCESS Framework which is designed to achieve these two related goals.

4 Open Source and the Older Adult

4.1 Introduction

There is very little literature available investigating the implications of providing open source software for older and novice users. Open source software is not simply the same as commercial software that is provided for free. The process of open source contribution, whether it be software or knowledge, is an unusual and relatively novel approach to building information artefacts – and the nature of that process has numerous implications. This is an area that has particular relevance to this thesis as the tool discussed in chapters five and six is an open source tool in itself.

In this chapter, the dissertation will discuss open source development, the motivations of those who contribute to the process, and the implications of the process for older and vulnerable novice computer users. The software developed as discussed later in this dissertation is open source software, but where possible it has been developed in line with the recommendations for software development outlined below.

As a final note, the nature of the open source community is that much of its discussion and deliberation is done online, and in forums and discussion groups. Where possible, reference has been made to the academic and popular literature of open source to substantiate the argument, but it has been necessary to reference more transient resources such as web-pages to illuminate certain points of key relevance.

4.2 Open Source and Collaborative Development

At its simplest, an open source application is one in which the source-code that builds the application is distributed alongside, or instead of, the executable binaries that comprise the program itself (Gurbani, Garvert, Herbsleb, 2005). For more substantial definitions, there are several organisations that have claimed philosophical ownership over the term, such as the Open Source Initiative (OSI) who have layered several other principles onto the core definition (Kirschner, 2008). Others such as the GNU project have made the term part of a larger movement within software development known as the 'free software movement' whereby the openness of the product becomes the key driving principle (Chopra and Dexter, 2007).

It is the nature of the terms under which a software application is distributed (the licensing agreement) that defines the freedom that users have with the software. Some licences prohibit modification for profit, while others act virally and require the adoption of the licence in all derivative works. At the time of writing, the OSI have no fewer than sixty-four distinct licences documented on their website (http://www.opensource.org/licenses/alphabetical). Popular licences include the GPL (General Public

Licence), the MIT Licence and the Mozilla Public Licence (MPL). The software developed for this dissertation is released under the Eclipse Public Licence (EPL). Kennedy (2001) discusses in depth some of the issues and complexities that come along with choosing the right kind of licence, but these issues have only tangential importance for non-technical end users.

The open source movement has attracted much ‘free’ labour to the cause of building open, transparent software systems. This has resulted in many substantial open source projects becoming reliable, scalable technologies that have been used at all levels of the digital economy, from individual servers to the hardware that runs mission critical systems for multinational corporations. Even so, some authors (for example, Gurbani, et al, 2005), have raised concerns about the suitability of open source as a process for developing commercial software. The largest and most successful of open source projects are community outcomes, resulting from the large-scale integration of the work of multiple contributors. The product of this contribution tends to follow the usual long-tail pattern whereby a small minority are responsible for the majority of content (such as is noted in Singh and Twidale, 2008; Sowe, Stamelos, Angelis, 2008). However, despite the peculiarities of this approach, it has resulted in many successful applications and frameworks that are used daily by millions of people - some of these include Apache, MySQL, Linux, Mozilla Firefox, Mediawiki, Perl, PHP and many others.

As part of the general philosophy of open-source development, a culture has emerged around this core idea – it binds tightly to a complementary system known variously as collaborative production (Benkler, 2002), crowd-sourcing (Brabham, 2008) or sometimes commons-based peer production (Nissenbaum and Benkler, 2006). Together, these can be taken as different expressions of a philosophy of open source content provision – this content can be software (such as Linux) or knowledge (such as in Wikipedia).

In systems such as these, individuals choose elements of a common resource to implement or improve, and the modifications are then folded into the original, often with some kind of central authority acting as an intermediary for approval (Raymond, 2001). Aside from a few core individuals, contributors to such projects do not usually receive any kind of financial reward for their efforts – instead, the context in which they operate more resembles that of a traditional gift culture (Raymond, 2001; Wu, Gerlach and Young, 2007) in which one’s personal standing is increased not by the amount an individual has, but by the amount an individual gives away. However, as a counterpoint to this, it should be noted that many of those who work most closely with the open source movement are remunerated as part of their job as advocates within larger technology organisations.

This collaborative approach to development has since extended beyond its roots in software source-code into areas such as collaborative knowledge creation, of which the most successful of these is the peer-produced knowledge resource known as Wikipedia. The remarkable success of Wikipedia as an engine for encouraging unremunerated participation has led several commentators to remark that we may be looking at an entirely new way of structuring society (Shirky, 2008; Tapscott and Williams, 2006). Whatever the validity of such claims, it is clear that collaborative production of resources has resulted in a great deal of concern for companies built on more traditional models of development (For some examples, see Harmon and Markoff, 1998; Raymond, 1998; Giles, 2005), and that the collaborative movement has boasted from the start many successful products competing directly with closed-source, proprietary alternatives (Moody, 2002).

4.3 Collaborative Participation and Remuneration

It is not immediately apparent why a process built on voluntary, unremunerated participation can result in serviceable products. Only a minority of individuals engaged in open source development are employed and financially compensated by the larger companies dedicated to open source initiatives (Lakhani and Wolf, 2003), and participants are giving their time and often marketable skill-sets to projects whereby any financial return accrues to other individuals or organizations. Several reasons for this have been proposed as a result of studies into motivation of open-source software development:

- Perfecting expertise (Moody, 2002; Raymond, 2001; Lakhani and Wolf, 2005; Von Krogh, 2003)
- Enhancing reputation (Bezroukov, 1999; Lakhani and Wolf, 2005; Raymond, 2001; Von Krogh, 2003)
- Fun and enjoyment (Von Krogh, 2003; Lakhani and Wolf, 2005; Moglen, 1999)
- Expectation of reciprocity (Raymond, 2001)
- Job Market Signalling (Raymond, 2001; Hars and Ou, 2001)
- Altruism (Nissenbaum and Benkler, 2006)
- Belief in a principle of development (Hertel, Niednar and Hermann, 2003; Raymond, 2001)
- Fulfil a personal need (von Hippel, 2001; Raymond, 2001)

All of these authors have stressed different motivations to different degrees. Most studies have treated the open-source phenomenon as a unified whole, but it does not hold that motivation will be identical across different domains of participation (Moore and Serva, 2007). Oreg and Nov (2008) discuss how the context of the collaboration impacts on the motivation of participants. These studies have focussed primarily on those who volunteer often specialised skill-sets to a project, and their results are not necessarily transferable to the more open participation of projects such as Wikipedia. However, studies on the topic of Wikipedia particularly have highlighted commonalities (Shirky, 2008; Hars and Ou, 2001; Anthony, Smith, Williamson, 2005).

The nature of open-source development is strictly meritorious in most cases – those central authorities that exist (Linus Torvalds (Linux), Jimmy Wales (Wikipedia), and Richard Stallman (the GNU project) to name a few) have risen to their positions largely as a result of their standing as initiators and sustainers in a project, and through demonstrated authorial credibility (Moody, 2002; Raymond, 2001; Reagle, 2007). Their ability to influence development persists as long as that credibility remains. Leadership is thus an on-going, emergent property of the process of development.

In cases where a leader loses this credibility, ownership of the project may be removed by 'forking' the project – a rival development team begins their own parallel development from a common code-base. Such actions are not usually looked upon positively in the open source community (Raymond, 2001; Moody, 2002), but they serve a purpose as a moderator of authority and a mechanism for ensuring multiple development directions can be pursued.

This notion of authorial leadership is embedded at all levels of the open-source development process. Individuals emerge as tribal elders as a result of sustained involvement with a project – a good deal of credibility is attached to people who have shown long-term commitment (Reagle, 2007). Individuals emerge as leaders as a result of their reputation amongst their peers, and that reputation is built through the work that is done. This constant peer-review of contribution results in an iterative revaluation and refactoring of software elements. This in turn leads individuals to perfect and hone their technical and presentation skills as successive improvements are made to their contributions, and they in turn make improvements to the contributions of others.

The expectation of generalized reciprocity underlines much of the nature of open-source participation. The expectation is not that there is a mapping between contributor and beneficiary, but that at some point the contributor will benefit from the work of someone else in the community (Wasko and Faraj, 2000; Hars and Ou, 2001). This ties into the principle of fulfilling a personal need – many software projects are developed because the author has a need for them (Raymond, 2001; Bonaccorsi and Rossi, 2003) or because the author seeks a particular creative outlet. These projects then get released to the larger development community, and the increased pool of interested parties result in further successive improvements being made to the software. Each party may be acting in their own self-interest by improving the software, but all benefit exponentially from the process.

The skill-sets that go along with open-source development also have implicit economic benefits for those individuals who work in technical fields. Participation in open-source developments provides a useful job-signalling role (Lerner and Tirole, 2000) indicating competence to potential employers, and allows individuals to demonstrate community standing when seeking venture capital. A body of freely available work acts then as a kind of signal for self-marketing (Hars and Ou, 2001).

Finally, many proponents of open source software feel a deep commitment to a philosophical principle. The Free Software Foundation (2008) puts it this way:

> *To understand the concept, you should think of "free" as in "free speech," not as in "free beer."*

This casts it as a moral issue of user empowerment. While Stallman may be an extreme example of open source proponents, many cite the commitment to the cause and community of open source as a motivator in and of itself (Hars and Ou, 2001; Lakhani and Wolf, 2005), and the ability of open source to act as a counter-balance to the considerable economic power of companies like Microsoft, Google and Apple is a draw for those who believe such power to be harmful to the industry (Lakhani and Wolf, 2005; Ghosh, 2005).

4.4 Open Source Participation

Having dealt with the issue of why people participate, the next issue to be addressed is how. The nature of open source development prohibits the use of a traditional, top-down management structure due to the exponentially increasing co-ordination and transaction costs (Benkler, 2002). Instead, open source projects are traditionally defined by three key principles:

- The ability of individuals to self-select their contributions (Raymond, 2001; Anthony et al, 2005)
- The ability of individuals to choose their level of participation (Raymond, 2001)
- The ease at which individual contributions can be integrated into the whole (Raymond, 2001; Moody, 2002).

The first principle allows for a high degree of efficiency in the allocation of individuals to tasks. In an open-source project, participants are not assigned a task to complete by the project leaders. Instead, they identify an area of the project in which they are interested, and in which they feel they have the competencies to make an improvement, and so they focus their efforts there (Raymond, 2001, Moody, 2002). This method ensures that no individual is given a task for which they have no competency, or are allocated a duty which they find personally onerous. Critics of the open source model claim that this leads to uneven distribution of attention across an application, whereby the high-profile, high-prestige tasks get disproportionate attention compared to the low-profile, low-prestige tasks such as application documentation. Additionally, there are concerns that while open-source as a method thrives in 'interesting' problem spaces, it has problems producing applications of a more mundane nature (Bezroukov, 1999).

The strength of this approach though comes in the heterogeneous makeup of the developer pool for open source applications. What seems to be an insurmountable task to one developer may be trivial to another simply because their skill-sets, experience and mental models are especially amenable to that scenario (Raymond, 2001). This principle is summed up informally by Raymond (2001) as Linus' Law: 'Given enough eyeballs, all bugs are shallow'.

Coupled to this is the ability of an individual to choose their level of participation in the project. An individual may choose to develop entire new features, or focus on fixing small bugs in existing features as they desire. Allowing highly granular levels of participation ensures that individuals are never allocated tasks that exceed their level of motivation to contribute (Tapscott and Williams, 2007). An individual may have the skill-set and the mental model needed to implement a specific piece of code, but they also need the will to do so. In order for an open-source application to support this, it has to permit people to participate at all levels of the process, from finding bugs to fixing bugs to suggesting features to implementing features.

The nature of this highly-granular self-selection process leads to an emphasis on the importance of integration. It must be possible for people to submit their changes and have them quickly incorporated into the main body of the code. The longer the period between submission and integration, the larger the likelihood of duplicated development effort as developers self-select tasks that have already been completed but not yet distributed. (Moody, 2002).

4.5 Ethical Implications of Open Source Software

All of this discussion of open source software raises a number of ethical issues with regards to introducing it to older users. The cause of open source software is sometimes driven by technical concerns, or is sometimes evangelical, or aimed at meeting niche requirements of those individuals who have a need for particular software solutions.

There are ethical implications in exposing older users to new software applications generally (Alm, 1992), and it is appropriate to spend some time considering the ethical issues that are raised from the discussion above with regards to incorporating open source into the provision of accessibility and cognitive support software.

First of all, open source software is provided 'as is' – technical support for the majority of open source projects is limited to what time the developer can spend addressing questions, and the collaborative help available through the use of online forums and discussion groups. Documentation in the form of manuals may be extremely limited, and erratically updated (Levesque, 2004). While older users who have internet access are usually comfortable with the use of email (Jones and Fox, 2009), most are less comfortable generally with the idea of using the internet for communication (Melenhorst and Bouwhuis, 2004) and the reliance on the online, informal support that tends to characterise the open source community (Levesque, 2005) puts them at a disadvantage. Such support can often be highly critical and combative, as is evidenced in the following two extracts from a random web-support forum:

> *Seriously people.*
>
> *If YOU cannot be bothered to do the simple and obvious task of Reading FAQs, SEARCHING for and reading related topics do... then do NOT expect US to answer you!*
>
> *If YOU cannot fathom that you are meant to supply details when you aska question - incuding the basics such as URLs... then do NOT expect US to bother attempting to help*
>
> (http://www.google.com/support/forum/p/Webmasters/thread?tid=029f6d5729b35a89&hl)

And:

> *I have read the FAQs and checked for similar issues: YES / NO*
>
> ^ *See that bit?* ^
>
> *That's for you to tell us that yo uactually bothered to read the flamming FAQs.*
>
> *Do you realise how many questions we answer every single day - that are Covered In The FAQs?*
>
> *It's ANNOYING!*
>
> *Not only are you wasting our time - you are forcing other peoples posts down/out of view!*
>
> *(those with real issues that actually tried to find the answers themselves / that aren't in the FAQ)*
>
> *And do NOT post Snitty little replies when you get called out on it!*
>
> *If you cannot be * bothered - do Not expect us to bother being nice!*
>
> (http://www.google.com/support/forum/p/Webmasters/thread?tid=2ee02f5dc3e536c9&hl=en)

Such posts are not uncommon, and while they reflect a perhaps understandable general frustration with those who display a lack of concern or understanding of netiquette, older and novice users run the danger of being on the receiving end of disproportionately aggressive and personal criticism for doing nothing more hell-worthy than asking a question on a forum specifically intended to ask for help.

Those packages that are well supported are usually done so on a service model where the product itself is free, but the technical support is billed. For an older user unfamiliar with new technology, this can be prohibitively expensive unless subsidized by an employer or other organisation. Coupled to this is the fact that open source software often suffers from usability issues (Nichols and Twidale, 2002; Raymond, 2001), and this creates a pair of linked problems that are likely to cause substantial complications for older users. This problem is exacerbated too by the fact that older adults are more likely to blame themselves than the technology with which they are working (Newell et al, 2006), and the nature of open source development is such that users are often expected to be critical of the software and actively part of the 'bug location' process (Raymond, 1999).

There are improvements being made to the usability of open source software, with projects such as GNOME usability working to increase the quality of the desktop experience for all users. These though are projects linked to either very successful open source projects, or open source projects within large and established corporations. The issue remains, especially with regards to hobbyist applications that have no substantial vendor support.

The open source development process lends itself well to distributed development along the lines of a core team being responsible for a core system, or perhaps several interlocking core systems, and hobbyist developers submitting their own expanded functionality in 'plug-in' form – this model is effectively employed in many open source software packages, where extra or enhanced functionality is available for download (the Firefox browser is one notable example, as is the Netbeans development environment).

For an open source distribution model that incorporates plug-ins, there are issues of trust and reliability of downloadable content. If a framework is open enough, there is no way a technically unsophisticated user can tell what any given plug-in is doing even if the source code is freely available. Some method of ensuring the trustworthiness of plug-ins is required to ensure that advantage is not taken of vulnerable members of the user-base. Older users in particular are less likely than younger users to trust the information they find online (Fox, 2005) and so steps must be taken to build credibility and sustain that credibility. Ceding a little development efficiency in exchange for a centralised repository of 'trusted plug-ins' is an effective mechanism for limiting the possibility of damage. Other technical solutions, such as running plug-ins in a sandbox (a security measure used to limit the access of a piece of software to certain sensitive parts of a running system) are inappropriate with regards to accessibility software which will, as a matter of course, need very low level access to the operating system in order to perform evaluative or corrective actions.

Other problems arise with regards to the core engine. If the core engine of an application is open-source, there is little to stop an individual making their own modified version of the engine and distributing it under the same (or even a different) name. Modern versions of the installation routine for the Bittorrent application contain the following warning:

> *A number of websites have created their own versions of the BitTorrent client and plug-ins that attempt to charge money for the applications and infect your computer with malicious code. To protect yourself, be sure to only download our software from www.bittorrent.com or one of our authorized distribution partners: www.download.com, www.sourceforget.net, www.tucows.com, and www.jumbo.com.*

If such software is downloaded in good faith from a disreputable site, then a user's computer can be entirely compromised with no easily identified outward signs of infection (for some example, http://news.bbc.co.uk/1/hi/technology/7907635.stm, http://www.eweekeurope.co.uk/news/web-users-prone-to-fake-av-2-13384), or users can be scammed into believing that they are liable for criminal damages (as in http://torrentfreak.com/malware-extort-cash-from-bittorrent-users-100411/), or strong-armed into buying software to fix purported malware infections.

A solution should also incorporate a strong degree of control over authorisation of new features, and also on distribution of the application itself. There are issues that are introduced as a result of this due to the provisions of many open source licences, most of which stipulate that any user is permitted to modify and distribute the software provided there is appropriate attribution. While it cannot be expected that a disreputable developer will necessarily honour all the protections implicit in these agreements, an open source licence of this nature cannot be used by the software if it cannot ethically honour all the provisions of that licence. Additionally, individuals are always free to fork an open source project if so desired, meaning control of the source-code can never be a guaranteed right.

One mechanism that has been shown to work is to distribute the code framework under one name, and trademark a brand name for the operational version, as was done for Google's Chrome web-browser. The code for Chrome is freely available, but Google circumvents the problem of multiple versions of their application being released by placing the brand name under trademark protection. This gives a legal recourse in the event of rogue developers appropriating the code, but the financial burden that is implied by aggressively protecting such a trademark puts it beyond the resources of most hobbyist developers.

Licences only protect as long as they are honoured - the open source 3D modelling software known as Blender has been repackaged and sold on (http://www.blender.org/blenderorg/blender-foundation/press/re-branding-blender/), which is within the rights of the licence. However, it has been sold on without attribution, and this is not permitted by the GPL under which the software has been released. Licences offer some measure of legal recourse, but cannot prevent abuse.

4.6 Designing Open Source Software for Older Users

A software application looking to make effective use of the features of open source development must then have a technical and social architecture in place that allows for these three features to be incorporated for developers:

- It must allow full access to the source-code, including documentation of standards and features.
    - Open source software development is predicated on availability of source code, and open standards are especially important when discussing accessibility.
- It must allow for users to report encountered issues and requests for features in some easily digestible and searchable format
    - Traditional mechanisms such as bug reporting via a forum, or making use of open source tools such as BugZilla are not appropriate for novice users. If possible, errors should be captured transparently, but this has implications for trustability as discussed in chapter ten.
- It must incorporate a temporally short loop between submission and implementation, or at least distribute the transaction cost of incorporating changes from many individuals through some form of delegation mechanism.
    - A well designed open source software development process allows for rapid integration of fixes and enhancements.

One example of such a process would be to provide a core engine of an application, and delegate the responsibility for managing open-source contribution to plug-ins that can easily be incorporated into the main application. In this way, only the engine must be centrally managed and responsibility for approving changes to each of the plug-ins resides with the developers themselves.

However, such plug-ins can be developed to malign ends in any software product that offers the low level interaction with the operating system that is required in order to provide accessibility functionality. As noted above, vulnerabilities exist with regards to individuals either hijacking an open source framework to distribute their own modified versions of the software, or taking advantage of the nature of open source distribution to produce their own, tainted plug-ins which they then attempt to distribute to users.

It is important in an application of this nature that users can be assured of the trustworthiness of the plug-ins they use, and so steps must be taken to ensure a central repository for any application code, and this in turn will permit a central authority for ‘signing’ plug-ins as being officially accepted. To ensure trustworthiness of the application, where possible it should be linked where appropriate to other trustworthy brand names.

It is relatively common for open source software to change dramatically from version to version, often with little regard for how it impacts on common routines of established users (one good example of this comes from when the GNOME windows manager software was switched over, by default, to using a new and unfamiliar metaphor for exploring a desktop, as discussed in http://www.osnews.com/story/7548). One quote from that article in particular is of relevance:

> *"'That's why the Gnome team chose to use the spatial mode as the default: expert users who want file hierarchies can change that within 20 seconds. Novice users, on the other hand, should never need to see a file hierarchy. It just makes sense this way.*

Unfortunately as has been discussed earlier, 'just makes sense' to a computer expert is not the same thing as 'just makes sense' to a novice user, and when new metaphors for interaction are changed by default because 'it makes sense', it frustrates the ability of users to understand and come to terms with new versions of software. As such, open source software being developed for older users must be mindful of the intended audience at all times. The following are simple principles that are important to ensure software is maximally relevant to the end users.

- The interface should remain consistent between versions of software. Unless there is a specific and justifiable reason why an interface should change, it shouldn't change by default.
  - Support for this requirement can be found in Colazzo, Molinari, and Tomasini, 2008, Ratvinder, Targonski, and Mach, 2008
- Functionality should always take second place to usability. New features should be introduced only to meet requirements and not to provide unwarranted 'richness'
  - This is related to the literature regarding the ever increasing complexity associated with learning software systems. (Stobie, 2005; Dickinson et al, 2002)
- Software should be consistent with other applications in the operating system context. Menus should be consistent with standard applications, and the look and feel of the software should be identical in so far as that is possible.
  - This is related to aiding older users to build consistent mental models for software interaction and task performance (Sharit, et al, 2008, Dickinson et al, Norman, 1988, Staggers and Norcio, 1993)
- Help communities aimed at these demographics should be strictly controlled and moderated.
  - This may require delegation of the responsibility for this to 'elder' members of the help community who have a track record for constructive engagement with novice posters.

An open source solution that impacts on older users then must provide compensatory strategies for these issues. The provision of high quality user documentation should be a priority of any tools developed, although the nature of open source makes it impossible for reliable technical support to be provided except at a cost to the developer. An open source application for older users should also include information on how to get more help using the software, and where that help is online and collaborative, a special effort should be made to ensure that the social environment is non-judgmental, non-threatening, and accommodating to reduce the chances of negative experiences and the resultant increase in anxiety (Todman and Drysdale, 2004).

Some of these principles go against the natural urges of software developers, especially those working in a highly iterative, collaborative development environment such as a good open source dynamic. However, they are important when dealing with software aimed at a generally inexperienced demographic such as older users.

An open source application designed for use by older users should incorporate the best practices of gerontechnological research (Zajicek, 2004; Newell et al, 2003; Newell et al, 2006; Dickinson et al, 2005; Hawthorne, 2000) in the construction of the user interface, and the user interface should stress consistency with other common applications.

4.7 Conclusion

Open source and older users is an area which is not well served by the literature on either gerontechnology or open source collaboration. There is a risk in introducing open source software into a context such as accessibility because of the high potential for possible harm – open source software, unless it has progressed to enterprise level and is in common usage, tends to be poorly documented, erratically maintained, and devoid of any of the usual channels of support and training that are associated with commercial packages.

Combined with this, any attempt to leverage open source development practises in the hope of providing a scalable approach to accessibility must take into account the risks posed by distributing effort across multiple developers – the open source development process works best when the transaction costs of integration are managed properly, but distributing such costs across, for example, development of accessibility plug-ins requires those responsible for the core software to be mindful of introducing potential vulnerabilities. It is the opinion of this thesis that open source tools have great potential in dealing with the accessibility needs of an increasingly computerised society. However, the context in which those open source tools are developed and distributed is vital when dealing with the vulnerable and inexperienced.

5 Overview of the ACCESS Framework

5.1 Introduction

This chapter will describe the design of ACCESS (Accessibility and Cognitive Capability Evaluation Support System), a software framework aimed at addressing many of the accessibility considerations raised in the previous chapters. The ACCESS Framework is designed with the following goals in mind:

- Lowering the participation threshold for those who wish to develop and deploy accessibility and cognitive support tools.
- Lowering the burden of knowledge required for users to configure accessibility and cognitive support on their systems.
- Leveraging the open source process to distribute the development burden over interested parties.

The ACCESS Framework takes the form of a core engine that is written specifically for an operating system context. That engine handles the low level communication with the underlying operating system through a set programming interface (known as an Application Programming Interface, or API). When individuals wish to write an accessibility tool, they can write it for the ACCESS Framework (in the form of a plug-in – a piece of software which is loaded into the engine) which then interprets the instructions for whatever operating system on which it is deployed.

The end result is a programming framework that provides cross-platform, cross system support for accessibility, while simultaneously reducing the burden of knowledge on individuals with accessibility and cognitive support needs.

5.2 The Design of a Framework

Programming frameworks are skeleton applications that provide the core functional capability of a system, freeing developers to concentrate on the elements that are unique to their own requirements (Johnson, 1997). Like a standard software library, a framework provides an API of which developers can make use. However, software libraries are passive and often require considerable investment of effort before a developer can profitably incorporate them into a bespoke application. In essence, a library provides a bank of functionality that a developer may choose to draw from.

A framework in comparison is active and the locus of control is inverted - a developer's application is incorporated into the running processes of the framework itself through the implementation of overloaded methods called at predefined points in the application's life cycle. Sweet (1985) refers to this as the Hollywood principle - 'Don't call us, we'll call you'. In order for this kind of system to function effectively, certain elements of functionality must be fixed (or frozen) to ensure cohesion of the framework. A framework empowers, but also limits what a developer can accomplish. As in all things, there are trade-offs that must be balanced in order to simultaneously meet the needs of the individual developer and the set of all developers.

The inversion of control that characterises a software framework has been identified as one of the fundamental techniques for code reuse (Gamma, Helm, Johnson, and Vlissides, 1995), and the role of a framework on reducing developer burden has been noted by Szypeski (2002, p159). Particularly in the field of accessibility where low level system events must be captured in order to provide accessibility support, there is an abnormally high burden of development experience required before even simple accessibility fixes can be deployed, and these low level system events can take on quite different forms across operating systems (as noted by Hanson et al, 2005). As such, even an accessibility solution that works well on one system must be rewritten to work properly on another, regardless of the language in which it is developed. Languages that offer platform independence do not offer the tight coupling with the I/O level that accessibility software often requires, and lower level languages rarely provide comprehensive tools for reusability and abstraction. A trade-off is thus required.

The ACCESS Framework is designed to remove the need for this trade-off by acting as a simplified virtual machine (See Lindholm and Yellin, 1999 for a treatment of the features of the Java Virtual Machine specifically) - plug-ins are developed to run on the ACCESS Framework, and a version of the framework can be implemented for each operating system. While the work of developing an ACCESS engine for a platform (known as an Operating System Context) is non-trivial, it only has to be done once and the functionality can be shared between all plug-ins.

The framework itself is designed with platform independence in mind, with those elements subject to change on multiple operating systems being incorporated within an abstract factory design pattern (Gamma et al, 1995, p87) for easy, context specific instantiation. The basic engine itself is written in Java to ensure high level platform independence. When the framework is loaded up, it checks to see what operating system it is currently deployed upon, and then instantiates the correct context for the current running environment. This is discussed in more depth in chapter six.

5.3 Platform Independence

The platform independence of the ACCESS Framework is achieved through the use of specialised implementations of operating system specific functionality. These implementations are accessed through an abstracted interface that defines the supported set of developer functionality. A plug-in will never directly access, for example, the windows registry (the database used by Windows to store system and application options). Instead, it sends a generic message (such as, 'write this information to the system settings') to the abstracted interface, which then executes the code for handling that operation on the current operating system. Thus, on a windows machine it will write the information to the registry, whereas on a Linux machine (depending on distribution) it may write to a particular flat file in a particular directory. The instruction to ‘get a list of running applications’ is handled one way in the Windows XP context (as a DLL file that can be invoked by the ACCESS context), and another in the Ubuntu context (by using the results of standard Linux commands that are piped into the framework), and another way still on OS X (through the use of Applescript). This way, the strengths of the individual platforms can be leveraged while still providing a consistent interface to the operating system.

Likewise, with low level system events, the provision of keyboard and mouse functionality is handled by a loader class that identifies its current operating platform and instantiates the appropriate platform specific functionality that hooks itself into the I/O streams. Again, each context can handle this differently – on the Windows XP context it is handled by a small daemon program that hooks into the event handing system, whereas OS X requires a more substantial program in order to function beyond the limitations placed on the system. This I/O class also normalises the provided data from the I/O streams for the framework, and then passes on normalised information regarding key and mouse events to each individual plug-in. On Windows, the clock() function in C++ gives the time in milliseconds, but on Linux this same function returns the time rounded up to the nearest thousand. To circumvent this, the I/O class has the responsibility for time-stamping input events as they are received, ensuring that the level of granularity on systems is consistent. Similarly, Linux and Windows use different formats for line-breaks, and so any input that involves multiple lines of text is first converted into a standard representation and then parsed before it is sent on to plug-ins. Thus, individual differences in representation across platforms are rendered into a consistent, 'access format' used by all plug-ins.

The end result of this is that each plug-in can be developed without thought given for the operational context, and the result of corrective actions will be comparable regardless of which system you are on. If a plug-in determines that the mouse cursor should be bigger based on the mouse data it is receiving, then that mouse cursor will be made bigger regardless of whether you are on Windows, Linux, or OS X.

For the software tool developed for this dissertation, focus was placed on a Windows XP context as that is the context with the largest impact in terms of older users. However, proof of concept developments showing the tool functioning across OS X, Ubuntu, and Windows 7 were developed to ensure that the theory outlined above would translate into practice.

5.4 Design of the Framework

Much of the operation of the framework should be transparent to plug-in developers. The only thing they need to be concerned with is the interface for their current operating system context (the API providing the tools with which they can interact with the system), and the required interface of a plug-in. Plug-ins must conform to the following interface, and each of these methods acts as a hook to the engine – the developer does not decide when these are called, they are called at predefined intervals by the engine itself. An outline of these methods is shown in table 1.

Method	**Description**
positiveReinforcement	Called when the user indicates they approve of a change that the plug-in has made to the system.
negativeReinforcement	Called when the user indicates they disapprove of a change that the plug-in has made to the system.
mouseMoved	Called every time the mouse moves. Plug-ins must maintain their own state regarding previous mouse movements.
mouseClicked	Called whenever the user clicks the mouse.
keyTyped	Called whenever a key is typed
Create	A constructor for plug-ins
wantsToMakeCorrection	If a plug-in believes it can make a useful corrective action, it should return true from this method.
performCorrection	Called whenever the engine selects a plug-in to perform a corrective action.
getName	Gives the name of this plug-in
Heartbeat	A general method called every second on plug-ins.
Tick	A method called every time the engine ticks over each of the plug-ins.
Reset	A method called every hour, and can be used for performing internal cleaning of state.
setState	Used to allow the plug-in to configure its own state.
getState	Used to query the plug-ins current state
saveMe	Used to save any data that should be persistent
loadMe	Used to load any data that should be persistent

Table 1- Interface for an ACCESS plug-in

A plug-in must provide at least a stub implementation of each of these methods in order for it to be compatible with the framework. Some methods are provided purely for convenience (such as the heartbeat, tick and reset methods), while others are required to provide meaningful feedback to the engine. Each plug-in is responsible for setting and manipulating its own internal state, and interpreting user corrections with regards to its internal trigger values.

5.5 The Design of Plug-ins

Plug-ins for the ACCESS Framework fall into two main categories: Invokable, and hidden.

Invokable plug-ins have some, or all, of their functionality packaged into an application that can be triggered by the user. A full example of this kind of plug-in will be discussed in the chapter on Electronic Notes. Such plug-ins are presented to the user in the form of a launcher pad, with the name of the plug-in clearly displayed on the button. This will be discussed in more depth in a later chapter, but all the issues that are associated with software applications generally are relevant to the design of invokable plug-ins.

The hidden plug-ins have no interface with which the user can interact. Instead, they interact with the user based on analysis of user input provided by the engine itself. A developer may be interested in providing accessibility support for mouse interaction, and so the plug-in they develop will register an interest in mouse events. Inside the plug-in it can perform whatever calculations that the developer feels are appropriate and then make whatever corrections that seem worthwhile on the basis of that analysis. The expressiveness of the framework is, at time of writing, limited to what was needed to support the plug-ins that were written as part of the exploratory research, but they are currently able to change pointer size, pointer speed, pointer precision, size of pointer trails, double click delays, keyboard de-bounce rates, and keyboard repeat rates. They can also capture user input and replay it back so that it interacts with user interface elements. They have access to low level data streams for mouse and keyboard, and can also query the current working application and user login information. Anything stored in the registry is available for future expansions of the system.

At regular intervals (ticks) through the running of the tool, it will poll the state of each plug-in, asking if it believes that corrective action is required on their part. Those plug-ins that register an interest in performing a corrective action are then placed in a roulette wheel[2] of weighted probabilities. The engine spins the wheel, and then calls a method in the selected plug-in that indicates it should perform whatever corrective action it feels is necessary. Upon doing this, the engine flashes up a message to the user, explaining the change that was performed (as reported by the plug-in), and provides an option for a user to 'like' or 'dislike' the change.

One potential weakness in the design of this engine is that since plug-ins themselves have responsibility for interpreting feedback, and also for performing and undoing corrective action, that they can simply ignore the user. It is possible to design around this problem by making the

[2] A roulette wheel is a data structure in which an element of a list is randomly selected with a probability based on its weighting. A set with element "a" (weighting 10), "b" (weighting 20), and "c" (weighting 30) would, over a long enough run, provide "c" half of the time, "b" a third of the time, and "a" the rest of the time.

framework take a snapshot of whatever values are changed (possible through the fact everything is mediated through the framework) and undoing them directly, but at the moment the flexibility resides with the plug-ins until such time as it ceases to be an effective mechanism.

5.6 Reinforcement Learning

At the core of the user interaction loop is a system of operant conditioning based on the work of Skinner (1957). Skinner's work describes four patterns of conditioning that can be used to influence the behaviour of individuals – positive reinforcement (application of a positive stimulus); positive punishment (application of an adverse stimulus); negative punishment (removal of a positive stimulus); and negative reinforcement (removal of an adverse stimulus). These patterns are outlined in table 2.

Type of Conditioning	**Example**	**Impact on Behaviour**
Positive Reinforcement	Animal is given food for performing a particular action.	Increases likelihood of behaviour.
Positive Punishment	Applying an electrical current to an animal for performing a particular action.	Decreases likelihood of behaviour.
Negative Reinforcement	Removing an electrical current from an animal for performing a particular action.	Increases likelihood of behaviour.
Negative Punishment	Taking away food when an animal performs a particular action.	Decreases likelihood of behaviour.

Table 2- Patterns of Operant Conditioning

Within the context of the ACCESS Framework, the conditioning is applied to individual plug-ins. Only two of these patterns are implemented – positive reinforcement, which is applied when individuals approve of changes that have been made; and positive punishment, which is applied when individual disapprove. All that is of relevance to the framework is whether a plug-in should be more likely to trigger, or less likely, and as such only the impact of behaviour matters. Since there are no problematic ethical issues with positive punishment in this context (as there would be if the conditioning was being applied to living creatures), positive reinforcement and positive punishment are the conditioning patterns that have been implemented.

When it comes time to choose a plug-in to make a correction, the Framework constructs a roulette wheel of the plug-ins that have registered an interest in making a change to the user's operating context. This wheel is constructed based on the weightings of each plug-in. The wheel is then spun to choose which plug-in is permitted to make a particular correction. Lowering the weighting of a plug-in then will reduce the chance it will be selected in any given wheel. Positive reinforcement increases the weighting, and positive punishment decreases it. Since plug-ins have only one behaviour (trigger when they encounter problems they can correct), the **likelihood** of this behaviour is captured in their weighted probability.

The opportunity to reinforce a plug-in is provided after the plug-in has made its correction to the underlying system – the user is provided with the information about the change that has been made and asked to either 'like' (positive reinforcement) or 'dislike' (positive punishment) the change. The option to condition remains in place for a number of ticks - if the plug-in has not been conditioned after the preset number of ticks, consent will be silently assumed, but no adjustments will be made to the internal weights of the plug-in. The flow of this is shown in figure 3.

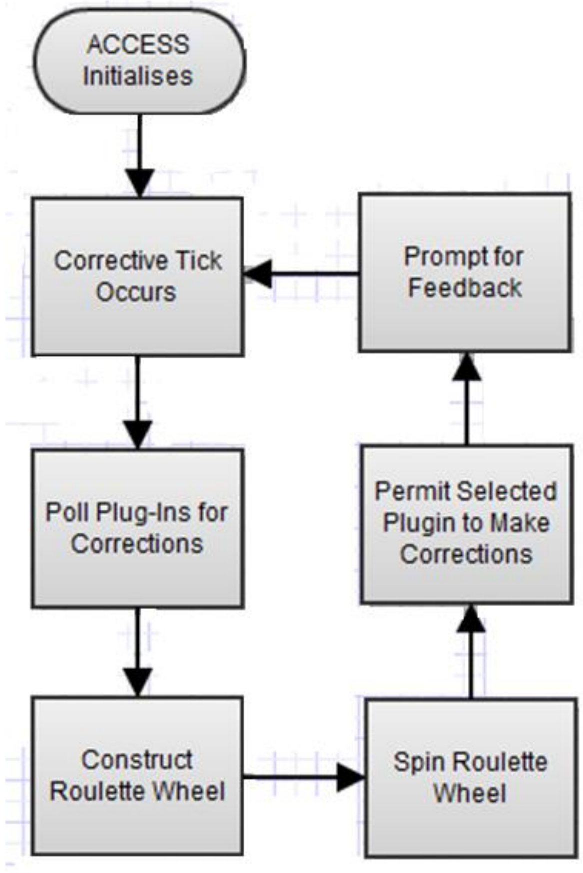


Figure 3 - Flowchart of ACCESS Lifecycle

When conditioning is applied, the Framework will also pass on details of the conditioning action to the plug-in itself. Plug-ins can then use this information to adjust their own internal values to make them more or less sensitive as is appropriate. A plug-in interested in detecting when a mouse is shaken for example can make its calculations more or less sensitive based on user feedback. An example of a plug-in (PointerSize) that makes use of this flexibility is discussed in chapter seven.

As user feedback is recorded over time, a user influences which plug-ins have impact on their operating environment by directly modifying the chance they perform corrective action, as is shown in figure 4:

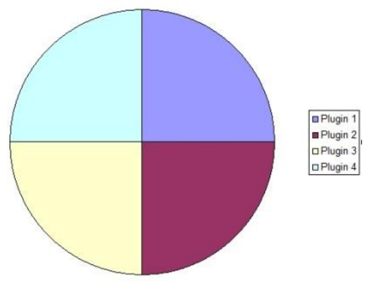
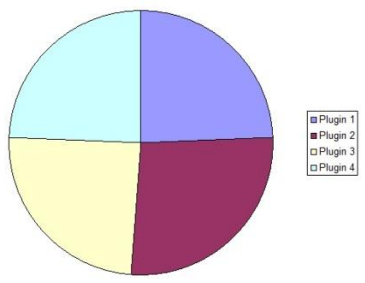
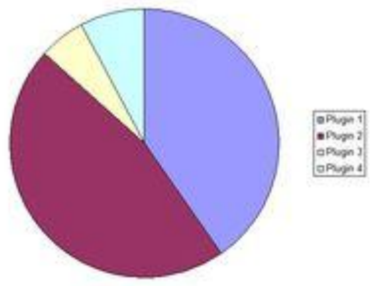
		
Starting state – all plug-ins are equally weighted, and all plug-ins equally likely to be selected if they express an interest in corrective action.	*After one reinforcement – one plug-in slightly more likely than the others.*	*After many reinforcements – two plug-ins very likely to be selected, two plug-ins very unlikely.*

Figure 4 – Sample Plug-in Weightings after Reinforcements

It is important to appreciate however that binary choices lack expressiveness. Core to this design though is one of the hypotheses that granularity of user opinions can be adequately assessed via an analysis of a series of binary data points. Repeated reinforcements for example result in greater impact on selection weighting for a plug-in, and internally a plug-in can provide whatever analysis is required for its own context sensitive reconfiguration. The framework however does not present the results of this analysis to the user – the user's only decision is to say whether they liked a change that was made, or they didn't.

The provision of these two options ensures tractability – users do not need to know the range of configuration options available to them, and they do not need to understand when an option is of value to them. The control panel of any modern operating system is an unknown quantity for many older users and in many cases they are unaware of some of its options entirely (see chapter three for a discussion of this). Upon being presented with a sample window from a relevant control panel, the user is presented with a combination of jargon, range of choices, and lack of clear guidance as to when an option has use to a particular individual. This ensures that configuring their computers for optimal accessibility is a task outside the comfort zone and knowledge of many of the individuals within this study. Thus, the goal in ACCESS is to remove as much of that complexity as is possible, rendering the decision down to a flat yes or no choice.

5.7 Plug-in and Application Communication

While the basic low-level daemons that interpret I/O information provide plug-ins with the majority of their input, the framework provides support for plug-ins to register themselves as listening on a socket. This allows for individual applications, such as the testing application developed as part of this project, to communicate with individual plug-ins and provide information that cannot otherwise be algorithmically detected. While it is not possible for the framework to simplify socket development on a client application, within the context of a plug-in the framework manages the multi-threading, instantiation and management of sockets and connections. All a plug-in must to do gain access to a socket is to implement a method that defines a port for the socket, and then implement a method to interpret the string data that comes in. For example, to register interest in having a plug-in listen on port 9999, the following method would be implemented in the plug-in:

```
public int createSocketPort () {
    return 9999;
}
```

And then to parse the information that came in:

```
public void parseEntry (String str) {
    if (str.equals ("miss")) {
        counter += 1;
    }
}
```

This allows for plug-ins to draw in application and context specific information to support their corrective actions. This is used in the missed clicks plug-in discussed later in the dissertation, and also within the input recorder (also discussed later) plug-in to allow for user input to be recorded and paused without the use of the GUI front-end.

5.8 Lowered Barriers to Participation

It seems unreasonable to expect that individuals who wish to provide accessibility support need to first navigate the low level I/O layers in order to develop even simple tools. Some programming languages make this easier than others, but modern virtualised languages such as Java and the .NET framework have increasingly abstracted away from the system layer. While this simplifies much programming, it makes it difficult for developers to act within an OS wide context without traversing some somewhat esoteric territory. Inexperienced developers are often faced with the prospect of developing outside of their comfort zone in order to implement even trivial accessibility functionality, and such tools are almost always developed in isolation without consideration given for what other equivalent tools they may be competing against for system resources and performing potentially costly operations.

Usually the actual 'core' functionality is at least moderately straightforward - the code for identifying mouse tremors involves calculations on points in a 2D co-ordinate space, and this can be as simple as performing arithmetic on the data elements in an array. It is not the accessibility functionality that takes up the greatest developer burden - it's the architecture needed for that functionality to be incorporated into the operating system.

Within the ACCESS Framework, the architecture for O/S communication is provided, and developer applications (the plug-ins) are completely passive - they receive information from the framework at pre-defined points of time and may or may not act upon that information as they choose. How a plug-in manages its own internal data structure is of no concern to the ACCESS Framework - a plug-in may be a single class, or a far more complex package of interrelated classes. They can use all of the tools of OO design to properly package their functionality. All that matters is that there is an ACCESS class that implements a simple set of defined behaviours and beyond that point the plug-in acts like a black box. A plug-in could even potentially have a plug-in framework of its own and it makes no difference to the engine.

At the time of writing, the assumption is that plug-ins in the framework are limited to activating or deactivating existing features within the operating system, although functionality is available within the framework for plug-ins to execute commands on the underlying operating system (such as starting up external applications) and to communicate with external applications. The design goal of platform independence however complicates the issue of incorporating features that are not part of the framework - invoking a particular application would require either that the application be available on all supported platforms, or that the plug-in be made unavailable to unsupported platforms. The result of the latter would be an unfortunate fracturing of the possible pool of future plug-ins. Additionally, plug-ins at the moment have only three possible sources of input – keyboard input, mouse input and information obtained from external applications via sockets. These are the input streams that were necessary to support the development of those plug-ins developed for the research – future work on the framework will include an expansion of this set of input streams to include data from other sources as is discussed in chapter twelve.

5.9 Lowered Burden of Knowledge

So far, this chapter has focused on the technical benefits that come from having a consistent software framework for accessibility. However, this is a tool designed primarily for end-users as well as developers, and technical benefits are not sufficient to persuade anyone that they should use a tool. This section will discuss the design of the ACCESS Framework from the perspective of the end-user.

Trewin (2000) identified some of the key problems with existing accessibility tools, as outlined in chapter two. These fall primarily into two main categories – users lack the capability or knowledge to make substantial changes, or users lack the confidence to make the changes.

5.9.1 Lack of Confidence

As outlined in a previous chapter, older users often lack confidence in their actions, and will often blame themselves for faults which more correctly belong with a badly designed application. Many accessibility solutions have a subtle impact that cannot be immediately observed (changing a key de-bounce rate, or slightly altering the speed of a mouse), and the lack of direct, obvious feedback on what has happened is an additional factor in the lack of confidence individuals display.

The ACCESS Framework attempts to resolve this issue by reducing the question down to 'do you like the change that has been performed' – there is no need for complex analysis of potential impact, just a statement of preference as to any changes that have been made. Where those changes are invisible, the framework suggests that the user simply try out the change for a while - time is permitted to experiment with changes, and because the framework will silently assume consent after a certain amount of time has passed, in cases where the user cannot detect the impact of a change, they can simply allow the framework to commit the alteration without having to make a specific decision about something they cannot judge. As the framework's implications for user deployment are the primary focus of this dissertation, this topic is discussed in more depth in later chapters.

5.9.2 Lack of Knowledge or Capability

Many useful accessibility tools built into an operating system are left underutilized because users simply do not know they are there. The ACCESS Framework resolves this problem by moving the responsibility for detecting the need for accessibility support to the plug-in itself. Only when a plug-in determines that support is required will a change be made to the user's system. In this way, a user does not have to be concerned with how to access the functionality – the functionality is provided when a need is externally identified.

This also resolves the problems of those who may be operating without the capability of making a change – those who have mouse interaction problems may not be able to navigate to the appropriate part of the operating system because of their impairments (As discussed in Zajicek, 2005). As with those who lack the knowledge to make an accessibility change, the provision of support when the need is identified also improves the situation for those who cannot access the underlying functionality.

As with the issues regarding user confidence, this is a primary focus of the dissertation and as such is discussed in greater depth in a later chapter with regards to experimental data regarding the framework.

5.10 Other Issues

Many other issues have been identified with regards to accessibility for older users, and a brief summary of how these impact on the design of this tool follows.

The desire for non-stigmatising technology (Sokoler and Svennson, 2007) is supported by hiding the presence of many plug-ins, and also making available invokable plug-ins: users can mask the need for accessibility support by indicating that it comes along with useful invokable plug-ins (which may or may not have any specific accessibility support in mind) that they find useful. An example of a non-stigmatising invokable plug-in is discussed in the chapter on the Electronic Notes plug-in.

The impact of dynamic diversity on the accessibility profile of a user (Gregor et al, 2002) is supported through the positive reinforcement and positive punishment system at the core of the selection of plug-ins. The engine is designed to mould itself to the particular accessibility requirements of an individual over time. Individuals express preferences towards particular plug-ins, and plug-ins apply those preferences to fine-tune the corrective actions they perform.

5.11 Issues with the Framework

Technically the system is uncontroversial – it has been developed and tested across multiple platforms, and the core design of the framework demonstrates low structural coupling and high conceptual cohesion. Design patterns have been incorporated as and when they contribute to improvements in the software architecture. Individual plug-ins developed using this framework require theoretical justification of their design as would any accessibility tool, and later chapters of this dissertation will focus on such individual design. However, leaving aside the technical considerations, the framework does bring up several important ethical issues as discussed in the previous chapter.

The first of these lies in the issue of user trust - specifically, how much should users trust the engine? Studies show that older users in particular are prone to identifying problems in interaction with themselves rather than the computers on which they are working (Newell et al 2006) - under such circumstances, can the initial assumption of consent implicit in the ACCESS Framework be counterproductive? It is a violation of user norms for a system to make a change to the system and only then seek consent, and that is what the framework does – consent for changes is assumed and then confirmed. The issue here is in the honesty of provided feedback – it’s important in the design to not to anthropomorphize the framework in such a way that it appears as if the feedback is anything other than cool, dispassionate assessment of corrective activity. Earlier versions of the software used a somewhat quirky 'helper monkey' avatar called Mojo, but later iterations of the software discarded this avatar in favour of simple, unadorned dialog boxes.

The secondary implication of this framework as an approach is that it violates what Schneiderman (1997) has termed the 'internal locus of control'. The assumption of consent allows for users to first assess what actual impact a change will have before they confirm or deny it, but in order to do this the system has to assume responsibility over the change and the user’s operating context. One potential solution to this is to adopt instead a system of 'consent then reinforce' in which a plug-in would indicate its intention to make a change and require the user to sign off on it. The problem with this though is that users may feel uncomfortable in ticking off a change, partially due to a lack of confidence and partially due to unfamiliarity with the impact of changes to be made. Potentially useful changes may be refused, and potentially harmful changes may be accepted without the user appreciating the end impact. Even so, a variation of the framework that adopted this mechanic is addressed in chapter ten.

These two objections have a third factor that acts as an additional imperative for resolution - that of the power present in plug-ins. It is trivial within the ACCESS Framework to develop a comprehensive, hidden key-logger for example - one that can keep track of application context and mouse input. Of a necessity, the framework must expose information that can be damaging to privacy. While some mechanisms could be applied to counteract this - such as scrambling the actual values of keys pressed - they all carry with them a cost in terms of potential accessibility functionality. This impacts directly on the issue of trustability discussed in the previous chapter, and the conclusions drawn there with regards to the importance of ensuring the integrity of the tool and also the integrity of individual plug-ins.

The last issue to be considered is that of consent in general. Alm (1994) has highlighted the ethical issues implicit in designing accessibility software, particularly in this case with regards to obtaining informed consent. While it is not a primary focus of this research, the same framework could support disabled users - proof of concept plug-in implementations for ACCESS exist for the Dynamic Keyboard (Trewin, 2004) and Invisible Keyguard (Trewin, 2002). Thus, there are segments of the intended future audience for this framework who may not legally be able to give consent, which can complicate the ethical issues surrounding co-opting control over accessibility options. These plug-ins are not aimed at older users, but were the framework to receive wider use as an accessibility tool, a proper discussion of this issue would be required.

5.12 Conclusion

Developing accessibility software requires a disproportionate knowledge of programming considering the often simple functional requirements being met. The ACCESS Framework has been developed as a way to lower the barrier of participation with regards to individuals being able to develop genuinely useful accessibility and cognitive support tools.

There are substantial ethical and conceptual issues that an open source framework of this nature raises, and these issues need to be addressed as part of the justification for its use. Chiefly, the power of plug-ins within the framework is a cause for concern, partially due to the access they have to user interaction and partially due to the ease at which plug-ins can work transparently in the background. Plug-ins for nefarious ends can be developed as easily as beneficial plug-ins, and the trust a user places on the framework can lower the suspicion a user may have regarding unexpected intrusions. Deployment of this tool in its current form then requires a framework for approving and distributing plug-ins as is outlined in the previous chapter.

6 Platforms and Plug-ins

6.1 Introduction

Having discussed the architecture for the ACCESS Framework in the previous chapter, this chapter will focus on some of the implementation issues that are associated with building accessibility software and how the framework resolves these. Each of the plug-ins developed for this dissertation is also discussed in this chapter, along with the specific techniques they use to identify the need for corrective activity.

6.2 Virtual Machines and the Deployment Context

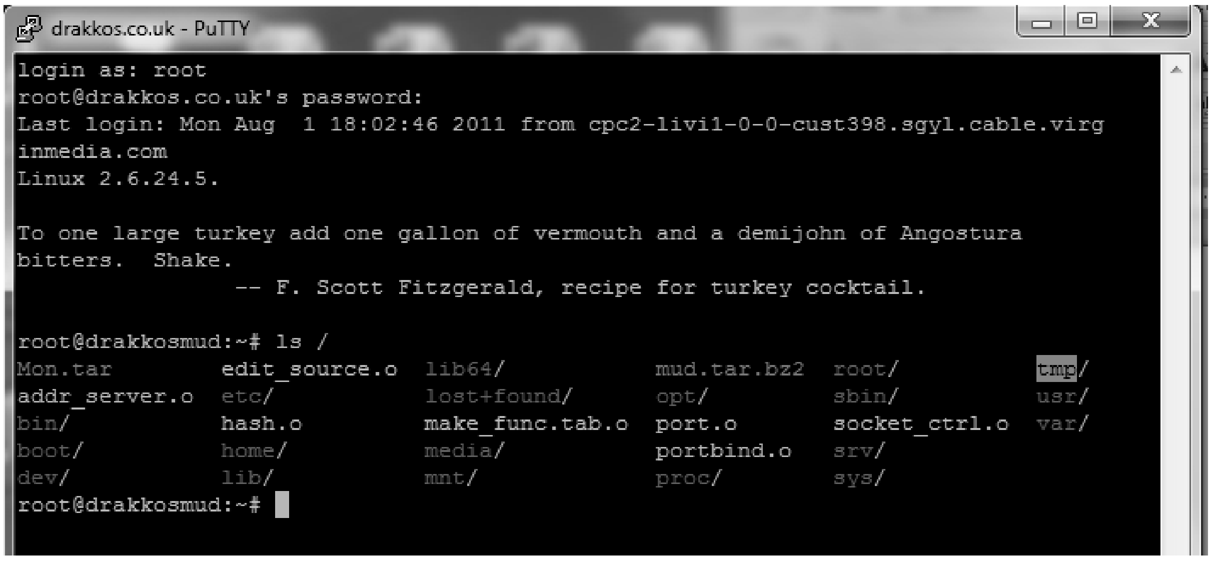
Abstracted programming languages such as Java and C# do not execute directly on the underlying operating system. Instead, they are compiled into a form of bytecode[3] and that bytecode is executed on a virtual machine. The virtual machine interprets this bytecode, and then makes requests of the underlying operating system to handle the functionality. This offers developers a tremendous benefit in that their programs become **platform independent -** provided an operating system has a supported virtual machine, the bytecode will run without modification.

However, the nature of this architecture means that developers are writing software for the virtual machine, and this necessitates that programs have little interaction with the operating system. Most languages offer classes for abstracting platform specific interactions such as saving files or storing preferences, but these are generalised implementations and abstract the key mechanics so as to ensure consistency between platforms. The result of this is that only functionality that is commonly supported on deployment platforms can be made available via the virtual machine.

This is an important limitation, and the developers of such languages have also provided mechanisms for accessing platform specific features. It's possible through the System class on Java for example to execute any arbitrary command line instruction. This can be used to invoke installed applications or access shell tools and pipe their responses back into the system. However, when this is done platform independence is lost because of the need to correctly handle the complexities associated with operating system differences.

One simple example of this is in the directory structures on Windows and Linux based systems. The directory structure on a Slackware Linux installation is shown in figure 5.

[3] Byte-code exists somewhere between compiled code and human readable source-code. It is amenable to software interpretation, as well as further compiling, and this make its it an ideal format for virtual machines.



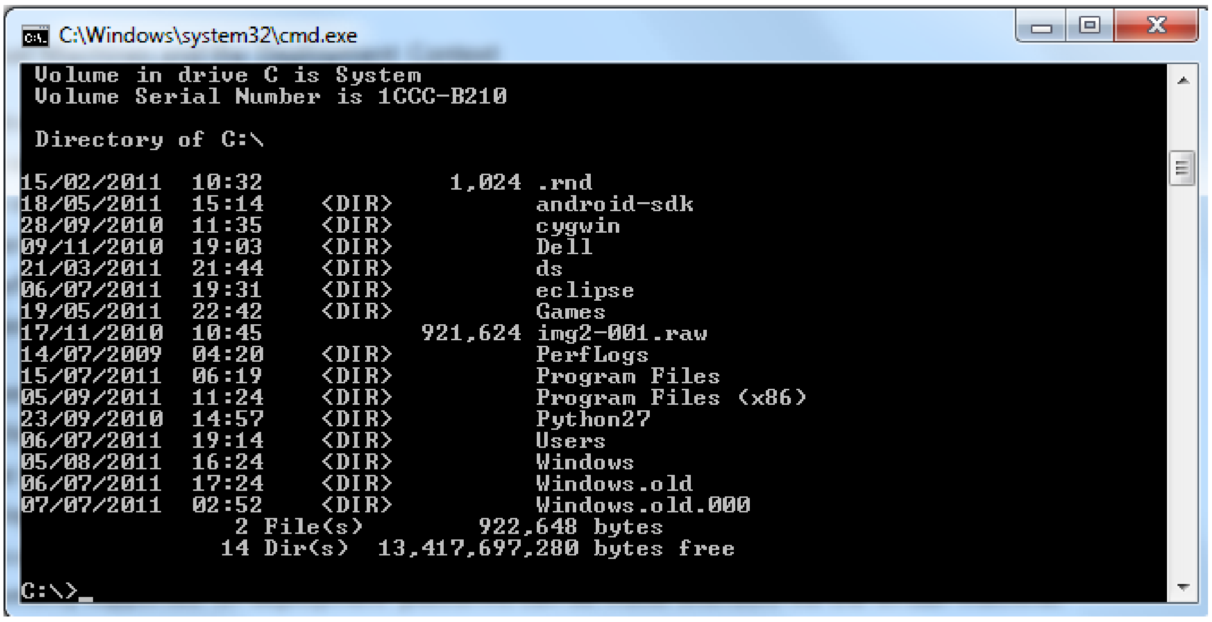
```
login as: root
root@drakkos.co.uk's password:
Last login: Mon Aug  1 18:02:46 2011 from cpc2-livi1-0-0-cust398.sgyl.cable.virg
inmedia.com
Linux 2.6.24.5.

To one large turkey add one gallon of vermouth and a demijohn of Angostura
bitters.  Shake.
                -- F. Scott Fitzgerald, recipe for turkey cocktail.

root@drakkosmud:~# ls /
Mon.tar         edit_source.o  lib64/           mud.tar.bz2  root/          tmp/
addr_server.o   etc/           lost+found/      opt/         sbin/          usr/
bin/            hash.o         make_func.tab.o  port.o       socket_ctrl.o  var/
boot/           home/          media/           portbind.o   srv/
dev/            lib/           mnt/             proc/        sys/
root@drakkosmud:~#
```

Figure 5 - The Slackware Linux Directory Structure

The Linux directory structure has a distribution specific role for each of these directories and subdirectories, with physical layout of drives being of secondary importance. The emphasis on Windows is that individual drives are the focus of directory management, as shown in figure 6.



```
 Volume in drive C is System
 Volume Serial Number is 1CCC-B210

 Directory of C:\

15/02/2011  10:32             1,024 .rnd
18/05/2011  15:14    <DIR>          android-sdk
28/09/2010  11:35    <DIR>          cygwin
09/11/2010  19:03    <DIR>          Dell
21/03/2011  21:44    <DIR>          ds
06/07/2011  19:31    <DIR>          eclipse
19/05/2011  22:42    <DIR>          Games
17/11/2010  10:45           921,624 img2-001.raw
14/07/2009  04:20    <DIR>          PerfLogs
15/07/2011  06:19    <DIR>          Program Files
05/09/2011  11:24    <DIR>          Program Files (x86)
23/09/2010  14:57    <DIR>          Python27
06/07/2011  19:14    <DIR>          Users
05/08/2011  16:24    <DIR>          Windows
06/07/2011  17:24    <DIR>          Windows.old
07/07/2011  02:52    <DIR>          Windows.old.000
               2 File(s)        922,648 bytes
              14 Dir(s)  13,417,697,280 bytes free

C:\>_
```

Figure 6 - The Windows 7 Directory Structure

Invoking an external application on Linux then is likely to require locating the appropriate binary in /usr/ or /bin/, and then executing it through the shell to start it up. The same task on Windows will require a different path because of the differences in the directory structures. As such, between Windows and Slackware, the developer must support two different directory structures in order to invoke an application even if we can assume that the same application is available on both systems.

Even within distributions of Linux, the structure of directories and the meaning of the structure have subtle variations. Each of these small differences must be managed.

A second, more substantial difference is in how different platforms handle storing of data files. Windows applications primarily function through database entries in the registry, whereas Linux applications use flat text files located in a particular directory. Thus, changing a particular system preference on Windows is mostly done through a consistent interface, whereas changing the same system setting on Linux requires locating the appropriate configuration file and making the adjustment to the text within. These are not simply philosophical differences – they have a major impact on the way in which applications must interact with the underlying system. The registry permits for values to be queried and set if you have access to the correct identifiers, whereas text-file configuration requires parsing the file in code and then committing the change to the underlying system.

Moreover, these systems are not consistent even within different versions of the operating systems themselves. The location of files can change between updates of a system, or be made entirely obsolete. Indeed, even different applications on the same platform can behave differently – this is particularly marked when Linux tools are ported over to Windows, since the philosophy of text configuration files often remains in place. These problems exist regardless of which kind of programming language is used – C and C++ do not have to mediate through a virtual machine, but in order to make these systems work on multiple platforms they must instead go through a manual process of 'porting' in which incompatibilities between one set of code and another are resolved.

Other issues are introduced when we as developers need to get access to parts of the operating system that the virtual machine does not support. Java and the .NET virtual machines do not permit access to low-level IO streams – if you need access to these, you must write in a language that is closer to the operating system that can be permitted through such virtualisation. The only way to get hold of keyboard and mouse input is to link it up to an active GUI window. The input is available only when that window is active, and when the window is minimised or invisible, the input data is no longer available. As such, getting access to user input across an entire operating system – a task which is core to many kinds of accessibility solutions – is something that cannot be done within the confines of the virtual machine. If these kinds of interactions are routinely required, then a virtual machine actually gets in the way of productive development.

However, where virtual languages do excel is in their flexibility for routine software development, and in the amount of support and expertise already available. Java and C# are languages that are routinely used for teaching programming, and almost all developers (novice or otherwise) will have some exposure to the way in which they work.

6.3 Non-Virtualised Languages

Non virtualised languages are compiled and executed directly on a host system, without the need for a virtual machine. This provides much greater access to the underlying operating context, but at a greater cost of portability between systems. All of the operating system specific issues outlined hold true for non-virtualised languages, but are joined by a number of additional ones. Virtual machines enforce users to work with common functionality because that is all that a developer has available. Platform specific development is often tied directly into the context and making it work on another is a challenging and specialised task.

However, non-virtualised languages have one facility that virtual languages don't – they can access underlying input streams. That which is impossible in Java (getting access to keyboard and mouse input across the entire operating system) is achievable in C provided you understand the way that the underlying event hook structure works as is demonstrated in the code extract in figure 7.

```
LRESULT CALLBACK LowLevelKeyboardProc(int nCode, WPARAM wParam, LPARAM lParam){
        KBDLLHOOKSTRUCT *keyboard = (KBDLLHOOKSTRUCT *)lParam;
        UINT code;

        code = keyboard->vkCode;

        switch(wParam){
                case WM_KEYUP:
                        KeyLog ((char)code, 1);
                break;
                case WM_KEYDOWN:
                        KeyLog ((char)code, 0);
                break;
        }

        return CallNextHookEx(NULL, nCode, wParam, lParam);
}

int main()
{
        MSG msg;
        HINSTANCE appInstance = GetModuleHandle(NULL);

        SetWindowsHookEx(WH_KEYBOARD_LL, LowLevelKeyboardProc, appInstance, 0);

        while(GetMessage(&msg, NULL, 0, 0) > 0){
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
        return 0;
}
```

Figure 7 - System Input Hooks in Windows

This small C program hooks a function to be executed every time a key is pressed or depressed on the user's system. It works transparently (albeit at some CPU cost) on the user's system, and it works regardless of what a user may be doing at the time. Systems like this are critical in ensuring that accessibility tools deployed on a user's system function at all times.

However, this particular program is extremely tightly bound to the Windows context – it won't work at all on a Linux system, and an entirely different solution is required for deployment on that platform. Figure 8 shows an extract from an open source tool called LKL (Linux Key Logger) which was used to create the key listening framework for test deployments of ACCESS on Linux.

```
void start_log(struct lkl *lkl)
{
       int pressed_shift, pressed_alt, status;
       unsigned char c, table[TABLE_SIZE];

       c = status = pressed_shift = pressed_alt = 0;
       bzero(table, TABLE_SIZE);

       if(ioperm(lkl->port, 1, 1) == -1){
           perror("ioperm()");
           exit(1);
       }

       if(ioperm(KEYBOARD_STATUS_PORT, 1, 1) == -1){
           perror("ioperm()");
           exit(1);
       }

       while(1){
              status = inb(KEYBOARD_STATUS_PORT);

              if(lkl->debug) fprintf(stderr, "c=%d ", c) ;

              if(status == 20) {
                  c = inb(lkl->port);
                  if (c < TABLE_SIZE){
                     if(table[c] != 1){
                         if((c == 42)    // LShift
                            || (c == 54))       // RShift
                            lkl->pressed_shift = 1;

                         if(c == 56)            // LAlt, RAlt (dang!)
                            lkl->pressed_alt = 1;

                         do_output(c, lkl);
                         fflush(0);
                     }
                     table[c] = 1;
                  }else{
                     if(lkl->debug) fprintf(stderr, "d=%d ", (c&127));
                     table[c & 127] = 0; //&127 delete the pair bit

                     if((table[42] == 0)      // LShift
                        && (table[54] == 0))    // RShift
                         lkl->pressed_shift = 0;
                  }
              }
              usleep(MSEC); //Don't freeze your system, dude :
       }
}
```

Figure 8 - LKL Keylogger Code

The C code required for each platform must be written specifically for that context, and it is this porting that is the heaviest cost in developing in non-virtualised languages. In both cases, deep understanding of the underlying platform is required to implement the functionality, and the task of porting requires deep understanding of both the original context **and** the new context. The burden of knowledge on a developer is considerable – it is not simply a case of changing function names or constant values. Windows has an event hook structure that you can override, whereas Linux doesn't have this structure and thus requires continual polling of the keyboard port.

6.4 A Hybrid Approach

The issues then that arise are as follows:

- Virtual machines have no access to low level I/O streams.
  - And these are critical for developing accessibility software of the nature discussed in this dissertation.
- Non-Virtualised Languages have a heavy burden of porting associated with them.
- Portability between systems in both cases is complicated by differences in system configuration.

The approach that has been taken in ACCESS is to create a hybrid solution. The majority of the framework is written in Java which allows for portability and tractability for developers. Those parts of the system that cannot be done in Java have instead been delegated to small daemon programs that are executed as part of the framework's functioning. Communication between these daemons and the framework itself is handled via socket communication. Sitting between these daemons and the framework is an object called the 'operating system context'. This handles the implementation of operating system specific functionality, as shown in figure 9:

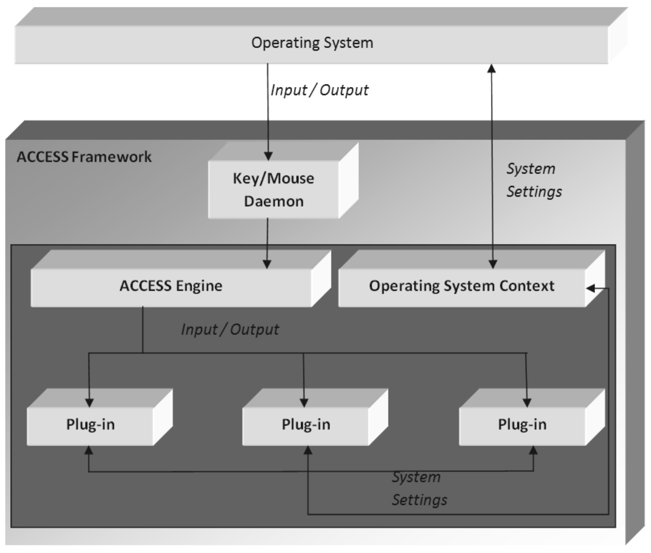


Figure 9 - Elements of the ACCESS Framework

Those parts in the light grey rectangle are tied into the operating system and must be written in an appropriate language (C for the code developed during this dissertation). Those in the dark are written in Java. JNI[4] is used where appropriate to handle the invocation of external functionality, but its use is confined to the operating system context.

To deploy ACCESS on a new platform, two things must be done – an operating system context must be written to implement the set of functionality supported by the framework, and a key/mouse daemon must be written and deployed. The task of doing these things may involve other programs being written, but these can be handled independently and without reference to the rest of the framework. As an example, to handle some of the Windows functionality, a DLL was written that handled things such as querying open windows and interacting with the registry. The registry stores values along with a set ‘type’ that defines how it should be accessed. In order to make this task as simple as possible for developers working within those darker parts of the framework outlined above, utility functions are deployed in the DLL to simplify this task as shown in figure 10.

```
void setStringKey (HKEY__* code, const char * subkey, const char * entry, const
char * type, const char *str) {
    HKEY hKey = 0;
    int buf[255] ={0};
    char aKey[1024], aValue[1024];
    DWORD dwType = 0;
    DWORD dwDisp;
    DWORD error;
    DWORD dwBufSize = sizeof (str);
    DWORD num;
    ostringstream tstr;

    if (strcmp (type, "REG_SZ") == 0) {
        cout << "REG_SZ" << endl;
        dwType = REG_SZ;
    }
    else if (strcmp (type, "REG_EXPAND_SZ") == 0) {
        cout << "REG_SZ" << endl;
        dwType = REG_EXPAND_SZ;
    }
    else {
        cout << "REG_DWORD" << endl;
        dwType = REG_DWORD;
    }

    cout << type << endl;

    strcpy (aKey, subkey);
    strcpy (aValue, entry);

    error = RegCreateKeyEx (code, aKey, 0, NULL, 0, KEY_WRITE, NULL, &hKey,
&dwDisp);

    cout << "Error: " << error << endl;

    cout << "Type is " << type << endl;

    if (error == ERROR_SUCCESS) {

        if (strcmp (type, "REG_DWORD") == 0) {
            cout << "Writing " << str << " as DWORD" << endl;
            num = atoi (&str[0]);
```

[4] Java Native Interface – a mechanism available in Java to use native code as if it was developed in Java. It is tremendously powerful, but it removes the portability of a java application.

```
            RegSetValueEx (hKey, entry, 0, dwType, (BYTE *)&num, sizeof (DWORD));
        }
        else {
            cout << "Writing " << str << " as SZ" << endl;
            RegSetValueEx (hKey, entry, 0, dwType, (LPBYTE)str, strlen(str));
        }
    }

    RegCloseKey(hKey);
}
```

Figure 10 - Windows Registry Access in C

This function in turn is used in a series of helper functions that remove the need for the developer to know the specific registry containers as shown in figure 11.

```
JNIEXPORT void JNICALL Java_accessframework_WindowsXpContext_setSystemKey
  (JNIEnv *env, jclass obj, jstring key, jstring entry, jstring type, jstring val)
{
    setStringKey (HKEY_LOCAL_MACHINE,
            env->GetStringUTFChars(key, false),
            env->GetStringUTFChars(entry, false),
            env->GetStringUTFChars(type, false),
            env->GetStringUTFChars (val, false));
}

JNIEXPORT void JNICALL Java_accessframework_WindowsXpContext_setUserKey
  (JNIEnv *env, jclass obj, jstring key, jstring entry, jstring type, jstring val)
{
    setStringKey (HKEY_CURRENT_USER,
            env->GetStringUTFChars(key, false),
            env->GetStringUTFChars(entry, false),
            env->GetStringUTFChars(type, false),
            env->GetStringUTFChars(val, false));
}
```

Figure 11 - JNI Helper Functions for Java

These helper functions are exposed in the operating system context as JNI method as shown in figure 12.

```
private static native void setSystemKey(String key, String entry, String type,
String value);
```

Figure 12 - JNI Method as Accessed in the Framework

The complexity for Java developers is confined to these native methods, meaning that the otherwise complex task of altering and querying registry settings is as simple as the following pair of functions.

```
     public int getMouseSpeed() {
        return Integer.parseInt(getUserKey("Control Panel\\Mouse",
"MouseSensitivity",
                "REG_SZ"));
    }

    public void setMouseSpeed (int val) {
        setUserKey("Control Panel\\Mouse", "MouseSensitivity", "REG_SZ", "" + val);

        refreshRegistryAlt(SPI_SETMOUSESPEED, SPI_SENDCHANGE, val);
    }
```

Figure 13 - Querying and Setting Registry Methods in ACCESS

Due to the complexity of the underlying representation of the registry, developers must also be aware of the various constant values used to indicate parameter changes (those used in refreshRegistryAlt), and the specific kind of method that should be used to commit the change to the underlying operating system. However, this is as complex as development within the framework gets, and it is limited to those who wish to expand and extend the expressiveness of the feature-set. New options can be added to the framework in a matter of minutes due to the high level of abstraction between the framework and the underlying DLL files that handle the complexities of configuration.

It is not then that the framework removes the need for developers to understand the inner workings of the registry, or that it doesn't require the use of pointers or system input hooks. It’s that these parts of the system are kept entirely separate from the parts that an accessibility researcher would be working with on a day to day basis. While novice programmers would be limited to the expressive potential provided by the framework, they can work productively within these constraints to accomplish their goals.

All of this must be implemented to create a deployment of the framework on a new operating system, but sometimes this task is simpler than requiring a complex interrelationship of different objects. Much can be accomplished using Linux shell commands and this obviates the need of extra coding – Windows XP requires a DLL file to get the current widow title, but it can be done on Linux and Max OS through the use of the **xprop** command line tool. Much of the complexity of changing system preferences on the Mac likewise can be handled through the use of applescript. In this way, each context can make use of the underlying platform. The framework puts no requirements on how particular functionality is to be implemented, and so it can be done using the best tool for the job.

6.5 Plug-Ins

None of this is of relevance to those who simply want to create an accessibility tool using the framework. The expressiveness of plug-ins is dependent on what is supported in the framework, but making use of that functionality involves simply making use of the facade provided by the operating system context as shown in figure 14.

```
public void adjustCursorSize(int intended) {
        switch (intended) {
            case 2:
                getEngine().getContext().makeCursorBigger();
                break;
            case 3:
                getEngine().getContext().makeCursorBiggest();
                break;
            default:
                getEngine().getContext().makeCursorNormal();
                break;
        }
    }
```

Figure 14 - Accessing the Operating System Context in a Plug-In

A plug-in requires no knowledge of which context is being used – it just requires that the context implements the methods that are defined as part of its facade.

Within this research, nine plug-ins were written, although only seven of these were experimentally trialled. The expressiveness of the framework was limited to what was required to implement the full set of plug-ins is shown in table 3.

Plug-in	**Description**
Missed Clicks	Made use of the socket communication framework to have the testing application discussed in chapter eight adjust pointer precision when a click missed the target.
Double Clicks	Tracked double click delay to dynamically adjust the double click threshold.
Pointer Size	Tracked difficulties in mouse location and then adjusted the pointer size accordingly
Mouse Trails	Tracked difficulties in mouse location and then adjusted the mouse trails accordingly.
Double Back	Detected ‘double back’ behaviour of the mouse and adjusted mouse speed accordingly.
Electronic Notes	Tracked which application was active, and adjusted the contents of an on-screen post-it note accordingly. This is discussed fully in chapter seven.
Input Recorder	Logged all user mouse and key interactions, and provided an interface for these to later be played back.
Invisible Keyguard	An implementation of Trewin’s (2002) tool. Discussed in cited paper. This plug-in was not experimentally trialled.
Dynamic Keyboard	An implementation of Tewin’s (2004) tool. Discussed in cited paper. This plug-in was not experimentally trialled.

Table 3- Table of Developed Plug-ins

Each of these plug-ins makes use of the framework as described above to handle operating system interactions, but also implements its own internal mechanisms for identifying when these interactions are required.

6.5.1 Missed Clicks

Several of the tasks described in chapter eight involve clicking on moving targets. The testing application included a mouse event so that when a click was registered that wasn't the target, it would send an inform to the framework via a socket. This information was then passed on to the missed click plug-in which then incremented an internal counter by one. When the counter reached ten over the course of the exercise, the plug-in would enable pointer precision as is shown in figure 15,

```
private int threshold = 10;
private int counter;

public void positiveReinforcement() {
    }

    public void negativeReinforcement() {
        getEngine().getContext().setPointerPrecision (false);
    }

public boolean wantsToMakeCorrection() {
        if (counter >= threshold && getEngine().getContext().getPointerPrecision()
            == false) {
            return true;
        }
        return false;
    }

    public void performCorrection() {
        getEngine().getContext().setPointerPrecision (true);
        counter = 0;
    }
```

Figure 15 - Missed Clicks Plug-in

This particular plug-in is not cross platform because it is dependent on an external application, but it does show some of the flexibility of the framework even when confined to a single operating system context. The relatively complex tasks of pulling in user input, modifying system settings, and communicating via applications and plug-ins is all handled by implementing stub methods, and none of the code associated with any of these methods is complex. An understanding of object orientation and the set of exposed methods in the framework API is all that is required to make a meaningful plug-in.

6.5.2 Double Clicks

The double clicks plug-in keeps track of difficulties in registering a double click as defined in the operating system settings. When a click is registered, the plug-in makes a note of the time it was recorded. When a second click is registered, it makes a note of the time that was recorded, and then compares the difference between the two time periods. If the time elapsed is greater than a full second, then the plug-in interprets this as two intentional single clicks. If the time elapsed is lower than this but greater than the threshold for a double click set in the operating system, then it registers this as an intended but failed double click. Each time this condition is triggered, the internal counter for the plug-in is increased by one. If it passes a threshold of two, a correction is made.

When the first click is registered, a limit is placed on mouse-movement within the framework with regards to assessing whether or not a double click was intended. Movement of more than 10 pixels in any direction between the first and second click removes a pair of clicks from consideration as an intended double click.

The code to implement this is shown in figure 16.

```
public class DoubleClickPlugin extends AccessPlugin {

    private int current;
    private long lastClick;
    private int threshold;
    private int counter;
    private int offset = 100;
    private int motion = 0;

    public DoubleClickPlugin (AccessEngine e) {
      super (e);
      current = getEngine().getContext().getDoubleClickDelay();
      threshold = 2;
    }

    public void positiveReinforcement() {
        threshold -= 1;
    }
    public void negativeReinforcement() {
        current -= offset;

        getEngine().getContext().setDoubleClickDelay (current);
        threshold += 1;
    }

    public void mouseMoved(long time, int x, int y) {
        if (lastClick == 0) {
            return;
        }

        if (motion > 10) {
            System.out.println ("Reseting click");
            lastClick = 0;
            motion = 0;
        }
        motion += 1;
    }

    public void mouseClicked (long time, String button) {
        long currentClick;
        long delay = 0;
        currentClick = System.currentTimeMillis();

        delay = currentClick - lastClick;

        if (delay >= 1000) {
            lastClick = currentClick;
            return;
        }



        System.out.println ("Delay is " + delay + ", current is " + current);

        if (delay > current) {
            counter += 1;
            System.out.println ("Counter increased to " + counter);
        }

        lastClick = 0;
    }

    public boolean wantsToMakeCorrection() {
      if (counter >= threshold && offset > 0) {
          return true;
      }
      return false;
    }

    public void performCorrection() {
        current += offset;

        getEngine().getContext().setDoubleClickDelay (current);
        System.out.println ("Delay is now " +
getEngine().getContext().getDoubleClickDelay());
        counter = 0;
    }

    public String getName() {
      return "Double Click Plugin";
    }

    public String getMessage() {
        return "Double-clicking has been made a little easier to do.";
    }
}
```

Figure 16 - Double Clicks Plug-in

6.5.3 Pointer Size and Mouse Trails

Both of these plug-ins share a core of code. They only differ in what their intended corrections are when difficulties are identified. In both cases, they implement three metrics for identifying when users are having issues locating their mouse cursor:

1. When the user moves the mouse to the top left hand corner, this counts as a lost cursor.
2. When the user moves the mouse sharply from side to side, this counts as a lost cursor.
3. When the user moves the mouse in spirals around the screen, this counts as a lost cursor.

For occasional loss of a mouse cursor, these are quick and effective workarounds. All are however sub-optimal. The first solution suffers in situations where the resolution of the screen extends beyond the monitor bounds (as is often the case with incorrectly configured equipment), because the mouse cursor will often disappear off the screen entirely. The second and third strategies require the user to catch the movement of the cursor, and such movement is not always immediately apparent especially in situations where that motion may be obscured by other movement or noisy, low contrast backgrounds.

For more pronounced problems with locating the mouse, there are accessibility solutions available in the operating system itself.

1. The mouse cursor can be made bigger
2. The mouse cursor can be given a trail when it is moved
3. The mouse cursor can be made to 'pulse' when a particular key is pressed and held.

Two plug-ins have been developed to pick up upon users having difficulty in mouse location, analysing mouse movements to pick up on the three compensatory behaviours outlined above. They differ only in the corrective action they make. The first plug in, PointerSize, will increase the size of the cursor when its internal thresholds are triggered. It will increase the cursor size each time it encounters a problem until the maximum cursor size has been reached, at which point it will cease to present itself for corrective actions. The second plug in, MouseTrails, will instead switch on the mouse trails. If mouse trails are already enabled, it will increase the length of those trails until the maximum has been reached.

For method two, the plug-in calculates a pair of values to reflect user difficulties. Firstly, an array of co-ordinates is collapsed into an array of directions from previous to current and this array is then flattened down into an array that contains only direction changes. For example, in figure 17.

(2,3), (3, 3), (4,2), (5,1), (6,1), (7,1)	(East, Northeast, Northeast, East, East)	(East, Northeast, East)
Co-ordinates	**Directions**	**Flattened Array**

Figure 17 - Example of Point Normalisation

From the flattened array, each pair of directions is assessed in turn for ‘abruptness’, which is how far the current direction is from the previous. Northeast and East have an abruptness of one, while East and West have an abruptness of four. The abruptness of each pair of points is calculated. The algorithm for determining mouse shake then is:

- Accumulate co-ordinates across a sustained mouse motion
  - If there are less than 50 data points, discard – motion was too short to analyse.
- Flatten the array
- Add the abruptness of each direction change to a counter
- Calculate the percentage of co-ordinates in the original array that registered a direction change
- Average the abruptness over the entire motion.

- Multiply the average of the abruptness by the percentage of direction changes.

If from this algorithm we get a value over five, this is interpreted as a mouse shake within the plug-in. This algorithm was developed via observation of users during a number of pilot studies for the research.

Spirals are detected using a similar mechanic – the array of points to be analysed is first flattened, and then the distance between each pair of directions is calculated. The following algorithm was then used to identify a spiral:

- Get the distance between current and previous directions
- If this distance is counter clockwise, then the current direction is set to false.
    - Otherwise, if it is clockwise, it is set to true.
    - If it is neither, continue with the next pair of points.
- If the distance between two points is greater than 2 then continue with the next pair of points.
- If the last direction is the same as the current direction, add one to the internal counter
- Set the last direction to be the current direction
- If the counter is greater than or equal to four, then a spiral has been detected.

As with the shake detection, this algorithm was developed as a result of observation of users during the pilot studies conducted prior to the actual research.

The last of the metrics for identifying a lost pointer is much simpler – if the mouse cursor moves within 20 pixels of the top left hand corner, the lost cursor counter is incremented by one.

All three of these metrics are analysed, and if more than two incidents are encountered during a tick of the framework, the plug-in will register its interest in making a correction – increasing the pointer size for the Pointer Size plug-in, and enabling or lengthening mouse trails for the Mouse Trails plug-in.

6.5.4 Double Back

The double-back plug-in looks to determine if the last few moments of a mouse interaction were of a different average direction than the main body of the movement. It does this by separating out an array of co-ordinates into two parts – the head and the tail. The tail consists of the last ten co-ordinates of the movement, and the head consists of everything else. If either part of the motion has fewer than ten entries, this is discarded as being too short for meaningful analysis. If the difference between the two averaged directions is equal to or greater than one, it counts this as an attempt to double-back and this increments the internal counter accordingly. If the counter reaches five during a tick of the framework, then the plug-in will reduce the speed of the cursor to attempt to correct difficulties in target acquiring.

6.5.5 Input Recorder

The last of these plug-ins was the input recorder, and this functioned silently and invisibly in the background recording user system interactions. It did this as logs of input events rather than as a video, since in this way it could be made to ‘play back’ a user’s interactions and have them interact appropriately with the operating system. This allows for the logs of user interactions to be used to fine-tune the framework performance since it is possible to use actual experimental interactions and analyse the impact of changes made to the plug-in algorithms. Of course, once the behaviour of the framework differs from that observed during the trial it is no longer possible to precisely replicate a session, but smaller segments of input can be extracted to create small, ‘interaction vignettes’.

The plug-in works both as an invokable plug-in (this provides the control panel for loading and replaying user interactions) and as an invisible plug-in that starts recording on the basis of a command issued over a socket. All user input from that point on is recorded and saved to disk. When it is to be played back, the framework makes use of the Robot class of Java, feeding back the input data with the delays indicated in the logs. Within the experimental trials discussed in chapter eight, the recorder was started when the testing began, and ended when the testing session ended.

6.5.6 Electronic Notes

Electronic Notes is an invokable plug-in that has no corrective functionality built into it. The interface for the Electronic plug-in is very simple – a button that can be pressed to either display or hide the note window, and the note window itself. It is an invokable plug-in, and so it is available as part of the launch-pad of the ACCESS Framework as is shown in figure 18:

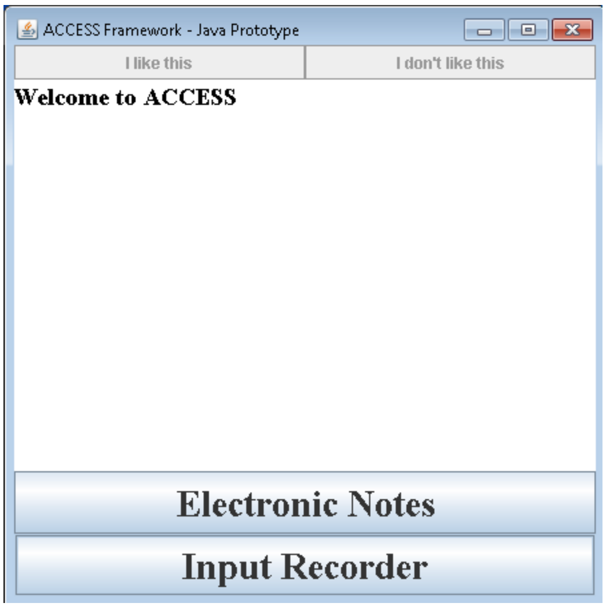
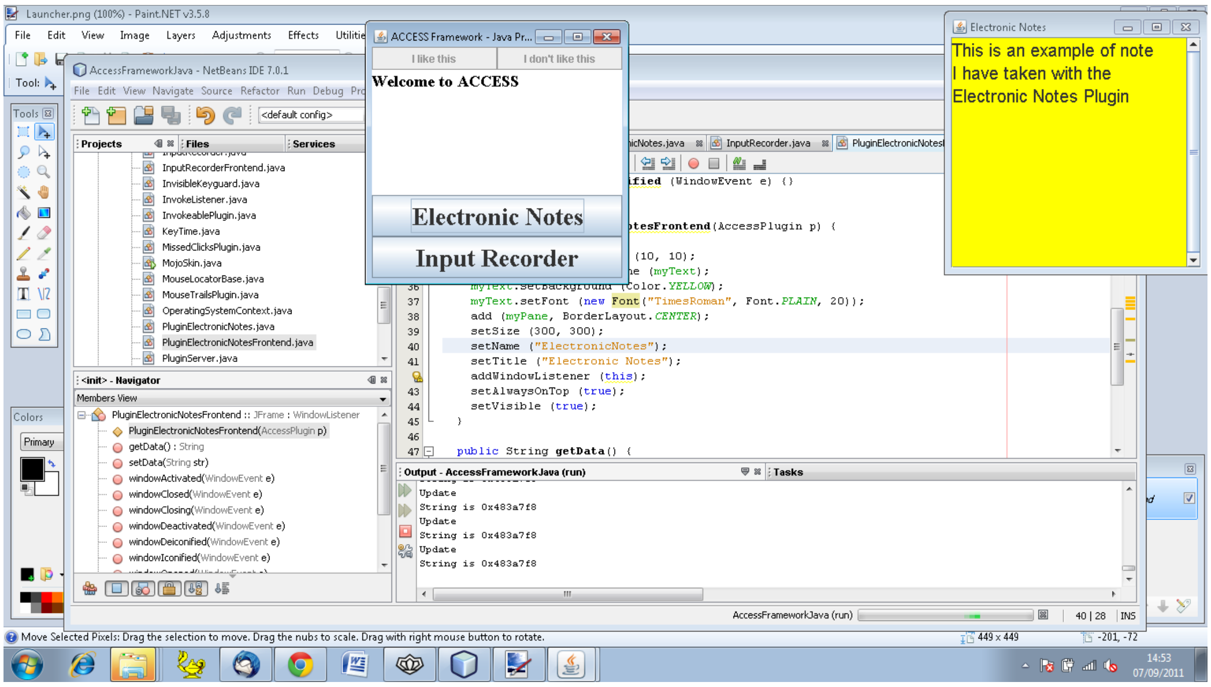


Figure 18 - ACCESS Window with Launch-pad

Figure 19 shows the plug-in after it has been invoked:



Welcome to ACCESS

This is an example of note
I have taken with the
Electronic Notes Plugin

```
ified (WindowEvent e) {}

otesFrontend(AccessPlugin p) {

 (10, 10);
e (myText);
myText.setBackground (Color.YELLOW);
myText.setFont (new Font("TimesRoman", Font.PLAIN, 20));
add (myPane, BorderLayout.CENTER);
setSize (300, 300);
setName ("ElectronicNotes");
setTitle ("Electronic Notes");
addWindowListener (this);
setAlwaysOnTop (true);
setVisible (true);
}

public String getData() {
```

Output - AccessFrameworkJava (run)

```
Update
String is 0x483a7f8
Update
String is 0x483a7f8
Update
String is 0x483a7f8
```

Figure 19 - Electronic Notes after Invocation

As the user shifts between applications, the notes available on the post-it interface will change to match the current context. Notes are persistent between invocations of the plug-in and executions of the framework. Formatting is retained, and the current context of the notes is indicated in the title bar along the top of the window. The post-it note will also behave like any other window, which means it can be minimised, maximised and hidden through the normal buttons at the top right of the interface. There is no corrective action or reinforcement associated with this plug-in – it should be thought of as a separate application which just happens to be making use of the cross-platform support of the framework. Electronic Notes is discussed more fully in chapter seven.

6.5.7 Other Plug-ins

The implementations of the Dynamic Keyboard and the Invisible Keyguard were developed with the intention of testing these on users, but the pilot studies showed that it was difficult to ensure difficulties with the keyboard tasks within the context of the intended user groups and the hardware provided. None of the participants in the study had physical disabilities that would have substantially impacted on their ability to use the keyboard, and those difficulties that they may have had could have been corrected by altering their physical proximity to the hardware. As such, these were written and trialled in the pilot studies, but never formally tested in the study. Full discussions of the techniques used in these tools can be found in Trewin (2002) and Trewin (2004) – no changes were made to the way in which the plug-ins functioned, the only work associated with these was to translate the techniques into a pair of ACCESS Framework plug-ins.

6.6 Conclusion

Virtual machines have enabled platform independence in a low cost way for developers. While their limitations do not impact on most routine application development, they are extremely problematic for accessibility developers due to the way they increasingly abstract the program code from the underlying operating system. The ACCESS Framework has been written to resolve this issue as outlined in this chapter – the primary technique used is a hybrid approach whereby platform specific elements are implemented in a non-virtualised language such as C, while the actionable code developed by individual accessibility researchers is designed to run in Java. The framework itself handles interrelating all of these parts, instantiating the correct context, handling socket based communication, and providing access to the Framework API to each individual plug-in. Plug-ins are thus reduced to being small, conceptually simple implementation of stub functions.

The plug-ins discussed in this chapter are addressed again in the context of the studies conducted, but are collected here to show some of the details of their implementation as they relate to the framework itself. A full code accounting of two of these plug-ins is included in the appendices – one of these is the Mouse Trails plug-in, which due to the range of behaviour it is identifying is more complex than other plug-ins. For comparison, the full code of the Double Click plug-in is also provided.

7 Electronic Notes

7.1 Introduction

In the previous chapter this dissertation outlined the ACCESS Framework and the different kind of plug-ins that can be deployed within it. In this chapter, the dissertation will discuss a plug-in aimed at reducing cognitive load on individuals through the use of context sensitive notations. Note-taking behaviour in older users is noted often in the literature (for example, in Bothin and Clough, 2010; Dickinson et al, 2002), and this particular plug-in was informed by that research and also by personal interaction with individuals within Dundee University's User Centre (Newell, Gregor, & Alm, 2006), a regularly scheduled drop-in centre for older users.

Electronic Notes (the plug-in under consideration in this chapter) is a variation on the simple electronic post-it notes that come with many operating systems. Users can invoke the plug-in to be provided with an editable textbox, and any notes made when interacting with the post-it is saved between sessions. The novel aspect of this particular application is that the post-it notes change as the user changes between different applications. Thus, the plug-in window shows only those notes that are associated with a particular program.

7.2 Why Electronic Notes?

The Electronic Notes application is aimed to provide a measure of cognitive support for computer users, specifically older users. The cognitive and psychomotor decline of older users is well documented (see for example Sharit et al 2008; Czaja, 2005) and this particular application is designed to provide support for three issues:

1. The need for users to switch between two sets of spectacles for near and distance sight (Hawthorn, 2005)
2. Decline in working memory (Salthouse and Babcock, 1991)
3. Certain cognitive impairments which can impact on the quality of handwriting. (Dixon, Kurzman, Friesen, 1993)

Currently when a user with different sets of spectacles needs to read their own handwritten notes, they must swap to their reading glasses, and then back to their other glasses when they return to the computer screen. When consulting notes on a regular basis, this adds a new measure of frustration to working with a computer system, as well as adding an additional interruption to work flow with the resultant increase in stress (Mark, Gudith, and Klocke, 2008). One participant during the research conducted for this dissertation made the following extra comment on his questionnaire:

> *I found it very difficult to turn my gaze away to the number:letter sheet, decode the appropriate letter (remember it) then re-focus on the screen. This may be due to wearing varifocal lenses which need to be precisely fixed on whatever I'm reading with [the] correct part of the lense [sic] (ONLY A FELLOW WEARER WILL UNDERSTAND THIS]'*

This difficulty however can be alleviated by ensuring that individuals have a mechanism by which they can make their own notes within the computer system itself.

Obviously most applications come with help systems, but these rarely meet the needs of older users (Vouligny and Robert, 2005). Handwritten notes offer several advantages over computer help systems - they are individualistic, they are written in a style readily comprehensible to the author, and cover only the salient points of importance. However, there can often be many of these in multiple different levels of details, for multiple different applications. Computer support for this can be provided by allowing users to make the notes for specific applications without requiring them to know where these notes are actually stored.

Likewise, older users may experience cognitive impairment that impacts their fine motor control, and thus the legibility of their handwriting (Yan, Rountree, Massman, Doody, and Li, 2008). Transferring these notes into a digital context allows for hardware and software assistive technology to aid users in making their own notes. There is a considerable body of research already focused on the topic of digital 'sticky notes' (c.f. Mistry and Maes, 2009; Klemmer, Newman, Farrell, Bilezikjian and Landay, 2001; McGee, Cohen, and Wu, 2000; Whittaker, Swanson, Kucan, and Sidner, 1997; Johnson, Jellinek, Klotz, Rao and Card, 1993), but none of these focus on reducing the cognitive burden by automatically switching the contents dependent on context.

There were two specific desires with regards to the feedback obtained from individuals in the study. The first was to analyse the effectiveness of the plug-in itself, and the second was to inform future development of plug-ins for the ACCESS Framework. Thus, the study was focused both on analysis of the plug-in and on requirements gathering generally.

There are some deficiencies with the system as it is proposed with regards to other studies done on this topic, namely that individuals are restricted to text and cannot store graphical data or annotate their existing screens. The core of this study is on the usefulness of notes generally - additional functionality can be provided later through iterative improvements to the plug-in.

The study was qualitative in design, looking to elicit useful feedback about the design and structure of the post-it note plug-in without reference at all to the underlying ACCESS Framework.

7.3 Design of the Study

7.3.1 Participants

Volunteers were solicited within the Dundee University User Centre (henceforth known as the DUUC), and six individuals were selected for the study. All individuals are over 60 years of age, and have some computer experience, but all self-identified themselves as novices. Users within the DUUC are most confident with web-based activities, and most have a grounding in the use of Microsoft Word. As such, the tasks were designed around applications with which individuals were likely to be most familiar. Inclusion criteria were that individuals were 60+ years of age and familiar with the windows operating system. Exclusion criteria were any substantial accessibility issue out with the scope of the experiment. Recruitment within this centre is done by appointment, and guided by the centre administrator so as to minimise the testing burden on any particular individual. Thus, while participants were not chosen according to any randomised process, they were selected without reference to any of the experimental outcomes.

7.3.2 Materials

It was not possible to perform the study in the User Centre itself – the study was concerned primarily with the utility of instructions with regards to paper versus electronic notes, and part of that study was the provision of unambiguous instructions regarding the tasks to be performed. Computers in the User Centre vary both in hardware (Macs and PCs primarily), and operating systems (Windows XP and Windows Vista being present on different PCs). Additionally, configuration inconsistencies between versions of software (Office 2003 versus Office 2007), and even different applications entirely (Open Office versus Microsoft Office) meant that it was necessary to provide a clean, consistent platform for the testing.

A laptop running a VMWare Virtual Machine was used as the testing environment. Within this virtual machine, a clean install of Windows XP was deployed along with the necessary applications for running the ACCESS Framework. Office 2007 and Internet Explorer 8 were chosen for running the study, primarily for the additional novelty these applications would provide when individuals attempted the tasks.

7.3.3 Procedure

Individuals in the study were requested to perform a set of tasks in both Microsoft Word and Internet Explorer. For Microsoft Word, participants were asked to complete one of nine formatting or editing exercises on a provided Word document. For the Word and Internet Explorer section, participants were asked to complete 20 specific tasks involving swapping between the two. The full text of the tasks can be found in appendix seven of the dissertation. The tasks requested were relatively well defined, but focussed on actions that were likely to be novel for the participants. Full instructions were given to all participants as to how the tasks could be completed, and these can be found in appendix six. These instructions were written by the researcher, and while they were part of the package sent for ethical approval, they were not formally vetted or assessed by any external party.

Participants were broken into two groups – three individuals were assigned to paper documentation, and three were assigned to documentation provided through the Electronic Notes plug-in. Those individuals assigned to the paper trial were given printed versions of the instructions. Those assigned to the electronic notes trial were provided the instructions only as text within the electronic notes window.

In both cases, individuals were informed they could take any notes they wanted as they went along, and also ask the interviewer any questions about the tasks or the instructions they had been given. Individuals were also informed that the instructions they had been given were only one way of approaching the tasks, and they should feel free to attempt them in whatever way with which they felt most comfortable. It was stressed in all hand-outs and in interviewer briefing that it was the documentation and the software tool that was under examination, and not the participants themselves. Tasks were broken into two categories – intra-application (working within Microsoft Word) and inter-application (working across Microsoft Word and Internet Explorer).

Individuals were given an hour to complete as many of the tasks they could while they provided whatever feedback they thought was worthwhile regarding the tasks, the provided documentation, and their experience with computers generally. At the end of each of these sessions, some mop-up questions were asked regarding the study and the documentation with which they had been provided. The talk-aloud protocol was employed throughout.

Data Gathering

Data gathered was primarily qualitative, although the application also recorded user interactions with the software:

- Times at which the plug-in was invoked
- Times at which information was entered.
- Changes to the provided data

The nature of user interaction with the software, as outlined later in this chapter, meant that the quantitative data was too sparse to be suitable for analysis.

7.4 Hypotheses

Several hypotheses were put forward at the start of the study.

- Hypothesis 1: The Electronic Notes Software will be a more effective tool for providing user documentation than an equivalent set of paper documentation.
- Hypothesis 2: The difference in effectiveness will be more marked when the tool is used to support work across multiple applications than it is when used to support tasks within a single application.
- Hypothesis 3: Users will prefer their own notes to the instructions provided to all users as part of the application.

Each of these will be addressed in a later section of this chapter.

7.5 Observations from the Paper Instructions Condition

The first three participants in the study were provided paper documentation, and asked to perform a series of tasks – the list of tasks can be found in appendix seven, and the instructions participants were given can be found in appendix six. No participant managed to complete all of the tasks, and most managed only three of the first set of tasks and one of the second set of tasks. Expectations of participant ability to complete the tasks was much greater than actual ability, as is consistent with software engineers without large amounts of exposure to working with older adults (Newell et al, 2006, Knight and Jefsioutine, 2002).

Individuals in the paper trials were given one document that outlined the tasks to be performed, and one document that outlined the instructions. The provision of these two sets of information led to some confusion as individuals mixed up the documents, or forgot about them almost entirely in one case.

Each of the individuals in the paper study had a different approach to using the instructions:

- Participant A followed the instructions to the letter, insofar as was possible.
- Participant B consulted the instructions initially and then largely left them to one side while experimenting around the task.
- Participant C regularly forgot about the instructions and had to be reminded they existed, but felt more comfortable with experimentation around the tasks.

Of these participants, one self-identified his need for distance and reading glasses, and noted that it was inconvenient sometimes to swap between the screen and the written word.

All individuals in the paper study discussed their problems with the use of jargon in instructions, and in the applications they frequently worked with – jargon in particular was highlighted as a major barrier to understanding for all participants, consistent with the literature (Goodman et al 2003, Newell et al, 2006), and individuals often did not associate a particular application name with its purpose. Participant C for example was asked to open up a Microsoft Word document, and instead opened up Internet Explorer. This made instructions that referred to the application directly less useful, as individuals could not necessarily easy distinguish between 'copy and paste in Microsoft word' and 'copy and paste in Internet Explorer'. When given the instruction 'open this document from the desktop', Participant C attempted to search for the document in Google.

Additionally, the idiosyncratic behaviour of some software applications was highlighted as an issue – one example which two of the participants had issues with were Microsoft Word's 'soft styles'. Mousing over one of the styles in the toolbar temporarily applies the style to the selected text. Since one of the tasks involved individuals applying styles to specific words, they often thought the style had actually been applied when it had not. This is particularly problematic for Participant A, who followed the instructions to the letter in so far as was possible, and yet failed to complete any of the tasks without assistance as a result of a lack of effective feedback from the application.

All three individuals in this study made note of the operating context of the software, indicating that they way the laptop was configured was noticeably different with the way their usual computers in the user centre were set up. Participant A and C indicated that the tactile feedback of the keyboard was an issue for them, and Participant B had issues with the resolution of the screen. These participants were asked during the study if they knew whether these issues could be corrected, and if they knew how to make the correction within the operating systems. None of the participants who raised these issues knew how it could be done or what features for aiding in accessibility were available as is consistent with Trewin (2000) and Hawthorn (2003). Some of the participants made an effort to adjust the application to suit their own needs, but these tended to be on a application basis rather than across the entire operating system – Participant B for example magnified a document in Word rather than changing the resolution in the operating system. Thus, participants had developed a compensation strategy for each of the specific applications with which they were familiar.

In each case, the participants tended to work with the application as it was presented to them. All three chose to work with windowed versions of both Word and Internet Explorer rather than make them full screen. This particular observation has implications for the design of the electronic notes application as will be discussed in the next section.

None of the participants in the study made any notes on any of the documentation they had been provided, although all indicated that they liked to make notes as they went along when they were working with computers.

7.6 Observations from the Electronic Notes Condition

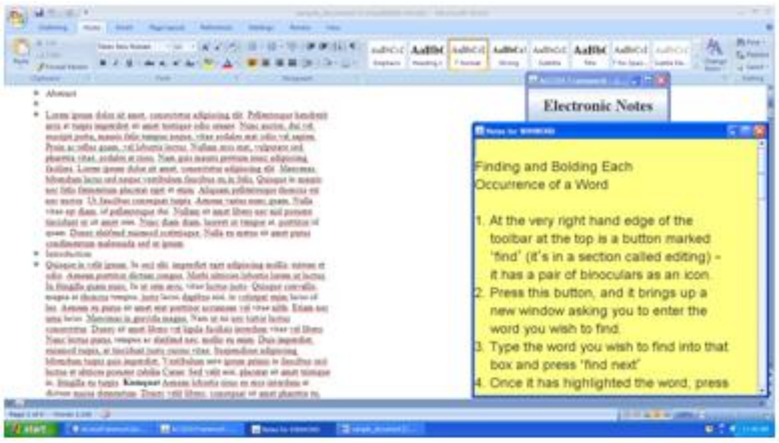
Individuals in this trial were given a short, five minute explanation on how the plug-in works and how it can be used. Consistent with the results from the paper trial, none of the participants in the Electronic Notes portion of the study managed to complete all the tasks, and performance was equally unsuitable for quantitative analysis across the two experimental groups.

Again, issues of inconsistency with their own preferred operating contexts was identified by participants in the Electronic Notes study. Participant D for example indicated that the provided mouse was much more sensitive than the one with which the participant was most familiar. Also, as with those participants in the paper study, individuals tended to work with the application as it was presented: while this was only an observational point for the paper based trial, it has considerable impact on the usability of the plug-in.

Older individuals have well documented issues with sight (as discussed in chapter two of this dissertation) and benefit more from lower resolutions - this has a knock-on effect to make every inch of screen real-estate more important. To provide instructions in a way that is actually readable, the electronic notes application must occupy a considerable amount of that screen real-estate. Even relatively modest tasks come with often quite intricate steps for an individual to perform. If the applications are used as presented without any attempt to optimise available screen real-estate, several issues are encountered:

- The plug-in interface can overlap parts of the application interface
- Users can unexpectedly shift contexts (mistakenly clicking on the desktop rather than on a button then causes the instructions to shift to those for the desktop, and clicking on a background application will cause the electronic notes application to shift context).
- Users do not make use of the option to selectively invoke and dismiss the post-it note window, and thus it becomes a permanent fixture of their desktop rather than a tool they have available when needed.

Optimization strategies for the screen real estate are possible, such as maximising the current working application (so as to avoid mistakenly clicking on the desktop), and positioning the notes application in such a way as it does not overlap any of the functionality of the application, as shown in figure 20.



Finding and Bolding Each Occurrence of a Word

1. At the very right hand edge of the toolbar at the top is a button marked 'find' (it's in a section called editing) – it has a pair of binoculars as an icon.
2. Press this button, and it brings up a new window asking you to enter the word you wish to find.
3. Type the word you wish to find into that box and press 'find next'
4. Once it has highlighted the word, press

Figure 20 - Optimised Windows Layout

However, configurations such as this require a disproportionate amount of knowledge to arrange, and represent a non-typical approach to a task workflow for the older users who participated in this study. Additionally, they introduce a new problem – specifically, when attempting to scroll through a document, the participant will often mistakenly scroll through the instructions, and vice versa.

However, the automatic context switching resolves one of the issues identified with paper documentation, which is that of associating an application with a set of instructions. Since that part of the process is done for the user, the user gets the correct instructions for the correct application provided they have that application window active.

Feedback from participants indicates that the electronic post it note application is too unwieldy given the often substantial amount of information that must be provided as instructions. However, in contrast to the paper documentation, none of the participants in the electronic notes portion of the study forgot that the instructions were available. As with the paper documentation trial, none of the participants in this portion of the study made use of the option to take notes or amend the instructions they were given, despite self-identifying that they liked to take notes as they worked with unfamiliar computer processes. Even in cases where clarification was given by the interviewer with regards to instructions provided, none of the individuals in the study modified the provided information in line with their own understanding of the process. An argument can be made that the issues encountered by individuals within this study may be 'centre-specific', stemming from the fact all participants were drawn from the same user centre. However, related research discussed in a later chapter revealed a large majority of users will self-identify as being someone who will work with an application as it is presented rather than attempt to optimize. This suggests that while centre-specific issues cannot be discounted from the analysis, it is unlikely that re-sampling with participants drawn from a wider catchment pool would yield more promising results.

However, despite the many downsides when working with the application as a way to provide complex instructions to individuals, feedback was generally positive about the potential of such a tool for bespoke note taking, especially if the possibility was available for sharing notes across systems. However, these comments must be assessed in line with the generally positive feedback older users tend to give with regards to software prototypes (Newell, Smith, Gregor and Dickinson, 2005).

Additionally, all participants in this part of the trial had only a few minutes of training on how to use the tool. Greater familiarity with the tool may reduce some of the more problematic issues, but it could be said that familiarity with any application would reduce the issues associated. As a tool that can be used 'out of the box' to support older users through providing an improved interface to pre-set instructions, the electronic notes plug-in does not offer promising results.

7.7 Discussion of Hypotheses

7.7.1 Hypothesis 1: The electronic Notes Software will be a more effective tool for providing user documentation than an equivalent set of paper documentation.

This particular hypothesis does not stand up in light of the results gained from observation of participants. Both sets of participants had equivalent performance in terms of completing the tasks, and both sets of participants indicated a range of issues with the instructions with regards to the complexity of the tasks and also the assumption of knowledge.

However, observational feedback of the two trials did indicate several areas in which the electronic notes plug-in provided additional benefits to users:

- Automatic context switching ensured that individuals did not become confused between different applications and sets of instructions.
- Provision of electronic notes ensured that individuals did not forget that the instructions were available.

However it also indicated areas in which new problems were introduced:

- Context switching could be unexpected when users mistakenly moved out of the application they thought they were working within.
- The necessity of the plug-in to provide comprehensive instructions meant it came at a great cost of screen real estate.

It is the conclusion of this thesis then on the basis of the qualitative feedback that the electronic notes application does not provide any substantial benefits to users when used as a platform for delivering externally provided instructions. However, it may prove useful as a mechanism for providing individuals to take their own, bespoke notes as they work with and between applications. There is no conclusion to be reached on the quantitative feedback as none of the individuals in the study managed to complete enough of the tasks to provide enough data for statistical evaluation, and the time-taken for individuals to complete even one of the tasks was so large as to prohibit any kind of data analysis on the results.

7.7.2 Hypothesis 2: The difference in effectiveness will be more marked when the tool is used to support work across multiple applications than it is when used to support tasks within a single application.

Due to the limited success individuals had in working through the tasks provided, there is no strong evidence either way regarding this hypothesis. Individuals struggled with both sets of tasks, and the limits that time placed on each of the studies meant that only a subset of tasks could be investigated. However, extrapolating from the issues individuals had when choosing between different sets of written instructions with regards to the same task in different applications, it seems that there is some small measure of support for the hypothesis that the plug-in's benefits would be most marked when working across multiple applications. Again, the lack of quantitative data means that the only conclusions that can be drawn from the study with regards to this hypothesis are qualitative.

7.7.3 Hypothesis 3: Users will prefer their own notes to the instructions provided to all users as part of the application

Users routinely took issue with the instructions provided, although each individual took issue with different parts of the information. Although the use of a fixed virtual machine ensured consistency of interface across each of the participants, the backgrounds, levels of familiarity and general attitude of each participant meant that they all had their own preferred way of approaching problems. This is consistent with the literature on electronic help which indicates that instructions provided for older users is often of very limited use (Such as discussed in Syme, Dickinson, Eisma and Gregor, 2003). One participant took issue with the very shallow nature of computer instructions generally, saying that they stopped people from understanding the link between their action and the outcome.

However, none of the participants in the trial actually made their own notes on the application, despite being encouraged in the initial briefing to do so. Evidence for this hypothesis then comes as an expressed preference for their own notes rather than anything linked specifically to the plug-in as it was deployed. Feedback from the electronic notes trials suggests that individuals could see a useful role for the plug-in in supporting their own note-taking behaviour. None felt that the plug-in provided any additional level of support with regards to providing pre-written instructions in a more accessible format.

7.8 Future Work

Future work on the Electronic Notes plug-in should focus on the tool as a support for existing note-taking behaviour and not as a vehicle for deploying electronic instructions. The tension of the trade-off between screen real-estate and usable instructions is such that it requires considerable window management to obtain maximal benefit. None of the participants in this study demonstrated any desire to manage their window arrangements, and one participant had no idea that it was even possible.

The discussion of the suitability of the tool as a mechanism for delivering instructions raised several issues that informed future development of plug-ins for the framework. By far the largest issue that individuals raised was that of jargon, and while this was not a focus of the research in this dissertation, it did inform the language choice of later plug-ins and is also discussed in the chapter on related and future work. Additionally, issues that particular individuals encountered as they worked their way through the tasks provided useful input into the design of other plug-ins aimed at addressing specific accessibility concerns in individuals. While some of these are out-with the current capabilities of the framework, they did serve as the basis for the design of the plug-ins that later followed.

7.9 Conclusion

Results from the study show that the plug-in offers no substantial benefits over paper documentation for older users, although results suggest that there are more promising avenues of research regarding its role in supporting existing note-taking behaviour. However, many of the problems associated with the plug-in are consequences of the general poor quality of computer instructions generally, and the overly complex work-flows that even relatively simple tasks require. For simpler tasks, many of the issues associated with the plug-in cease to be problematic, but the constant window management required to optimise screen real-estate is an unavoidable cost of providing the instructions in such a format. The benefits associated with the plug-in with regards to ensuring a correct link between instructions and context do not compensate for the additional cognitive burden imposed by the increased need for window management.

Feedback from users suggest that if the tool is used purely for resolving some of the issues associated with individuals taking hand-written notes of their workflows, then it could be a useful tool. However, conclusions on that matter are out with the scope of this particular study and would require further investigation.

8 A Study of Reinforcement Learning in Access – Design

8.1 Introduction

Core to the ACCESS system is the principle that reinforcement learning can provide an effective method by which individuals can configure their operating contexts to their own individual tastes. The system is wholly binary in user choices, which limits expressiveness and nuance but ensures the possibility of full comprehensibility of the ramifications of configuration decisions. Much of what is currently presented to users as configuration choices within settings menus and control panels offers precision, but at a cost of understanding – many users simply do not understand the full impact of their choices, even if they can penetrate the jargon associated with computer configuration.

This chapter will outline the main study of this research – an investigation into the effectiveness of reinforcement learning as a configuration regime for older users. This represents the key element of the ACCESS Framework as far as user interaction is concerned, and is thus important in assessing the applicability of the framework as a tool for end-users on their own computers as opposed to being something only of use as a development platform for accessibility researchers. Those accessibility concerns that are addressed by the framework are likely to be common across home and work settings, although in the workplace issues of user consent and satisfaction are likely to be secondary to corporate policy.

8.2 Research Questions

As discussed in section 5.6, within ACCESS, only two options are presented when a plug-in is selected to make a corrective action – 'I like this' or 'I don't like this'. Deciding on these options can be time delayed – the framework will not make another correction while the impact of the last is still being assessed, or until no decisions have been taken for a number of consecutive ticks. At this time the change is made but the weightings of plug-ins are not altered. The simplicity of this choice along with the immediacy of the perceivable impact is the key principle of the ACCESS user feedback loop.

This leaves the following research questions to be addressed:

1. Is it possible for an ACCESS plug-in to provide appropriate corrective action based on user reinforcement?
2. Is it possible for a combination of ACCESS plug-ins to interact in a consistent, holistic manner to produce a net improvement in an operating context?

A study was put together to address these research questions. For these, it is not important that the tasks are unrealistic and that they do not represent a realistic view of user interactions. The intention of the study is that users will encounter problems as a result of the context in which they are operating, and the framework in conjunction with its active plug-ins will be able to provide correction that is meaningful within the context of the study.

8.3 Participants

Through the SiDE project, individuals (N=38) of ages 65+ were recruited, with the intention of putting them through two separate repeated measures studies each. The gender balance obtained was M=15 and F=23.

8.4 Procedure

The experiment involved two separate studies, each lasting an hour including ten minute comfort breaks. The Double Click Study looked at the possibility for a single plug-in to provide appropriate corrective behaviour within a restricted context and tasks. The second looked at the interaction of five plug-ins across a wider range of tasks. At the beginning and the end of the study, each participant was asked to complete a questionnaire - the first assessed levels of confidence and familiarity with computing concepts, and the latter assessed their perceptions of the framework itself.

Each of these studies was aimed at assessing difficulties encountered by the user when using the mouse. An experimental study places a burden of time on the discovery of user issues, and so the context was intentionally configured to be as unfriendly to older users as is permissible within the operating system itself. This configuration did not involve external applications to artificially simulate difficulties. Everything in the configuration environment is something that is legitimately used by real users, and supported without extensions in the operating system itself. Even the choice of desktop wallpaper was restricted to options available by default in a Windows XP installation.

At the end of each task, the participant was requested to complete a small questionnaire that indicated how difficult they found the task, how responsive they found their mouse, and whether they had confidence that they could configure a computer to make the task easier in the future. At the beginning of each session, the ACCESS Framework engine reset the computer back to the initial unfriendly system settings.

8.5 The Tasking Software

All of the tasks required of participants in this study were made available through a piece of bespoke software that provided a set of buttons to invoke certain experimental situations. In addition, the software provided for testing schemes to be deployed - these schemes allow for instructions to be given between tasks, for the time between tasks to be varied, and for the recording plug-in to be started and stopped as a result of user interactions.

The set of tasks provided by this software were:

1. Double click a static target in the application window. Each click, the mouse cursor and the application window are moved somewhere random.
2. Double click a moving target in the application window. Each click, the mouse cursor and the application window are moved somewhere random. These buttons moved at the rate of two pixels every five milliseconds, meaning that they traversed the window in a total of three seconds from one side to the other. The target in this condition was forty pixels by forty pixels.
3. Click one of three static buttons that are placed randomly within a window. After each click, the mouse cursor and the application window are moved somewhere random. The button to be clicked is provided on a separate sheet of paper. Each button was seventy-five by 40 pixels.
4. Click one of three moving buttons that are placed randomly within a window. After each click, the mouse cursor and the application window are moved somewhere random. The button to be clicked is provided on a separate sheet of paper. The buttons moved at the same rate as the moving target in task two. Each button was seventy-five by 40 pixels.
5. Track a moving circular target around a window, keeping the mouse cursor as close to the centre as possible. This target moved at the same rate as the moving target in task two, and had a radius of 50 pixels.

At any point in any of the tasks, the user could choose to abandon it and move on to the next if it became too frustrating, and this was done by indicating their desire to the researcher. This was registered as abandonment in the system logs.

For each of the testing schemes, a single random number seed was used to generate all the tasks, and that seed was changed for each experimental condition. The result of this was that the behaviour appears random to a specific participant, but each participant would share exactly the same set of tasks down to the speed and direction of movement within particular experimental conditions. The tasking software maintained a log for users, indicating time delays between instances of a task, and any other actionable information. As with the ACCESS Framework itself, this software is available as open source.

8.6 Input Logging

One of the plug-ins that has been developed for the framework permits co-ordinate by co-ordinate and click-by-click logging of user input, and this plug-in was installed for each of the trials to capture user interactions. This set of input can be saved and loaded, and played back to reproduce the exact interactions that a user had with the study and session. This playback will interact appropriately with the operating system itself, so the collection of this data provides a reproducible set of real user data that can be used to fine-tune diagnostic behaviour in plug-ins, as well as serve as a mechanism for reproducing particular user interaction compensatory behaviour.

The fixed random number seeds in the tasking software ensured that it is possible to map user input study information onto the set of tasks they would be performing, to give a real-time replay of a user and their experimental session.

Having a corpus of user input in this way is tremendously valuable for working within the framework, as it allows for new interaction issues to be identified and then new plug-ins to be designed, developed and tested without requiring another potentially costly series of experiments to be performed. Part of the cost of developing a new plug-in is the need for the software to be written and then refined in the face of user experience. This corpus of testing data allows for software to be written and refined with real interaction data before it is tried out on actual users.

8.7 The Double Click Study

The Double Click Study looked at difficulties in performing a pair of double-clicking tasks (tasks one and two as outlined above). Companion to this task is an ACCESS plug-in that looks for double-clicking difficulty. It first records the current double-click threshold set in the operating system, and then uses a generous definition of an 'intended double click' to assess when a user was trying to perform that action. The definition used is 'two clicks made within a second of each other, with no substantial mouse movement between them', as is discussed within chapter six.

The tasks are first performed without any corrective software, and then again with the software in a repeated measures design – each participant then attempted each of the tasks twice. The tasks were counterbalanced to compensate for familiarity bias. After performing each task, the participant is asked to fill out a mini-questionnaire identifying their perception of how difficult they found it.

In this study, the focus is on whether or not a single plug-in within the framework can interpret user input to create a net improvement in both self-reported user satisfaction and objective measures of performance. The research questions here have several specific flavours that such a restricted context can answer:

1. Will an individual make use of the option to reinforce the plug-in appropriately?
2. Will a single plug-in be able to improve user perceptions of task difficulty?
3. Will a single plug-in be able to improve quantitative benchmarks for task completion?

Each run through the study then provides a measure of how quickly users completed the tasks, and their subjective perception of how difficult the task was. The following hypotheses follow then from this particular study:

8.7.1 Hypotheses

8.7.1.1 Participants will reinforce the plug-in

In order for the framework to correctly model user preferences, users must positively reinforce and positively punish the plug-ins that are selected in light of their own assessment of value. When a correction is made, the framework will present itself and a plain language description of the change that was made to the user for reinforcement. It is hypothesized that this prompting will encourage participants to reinforce the plug-in. For this hypothesis, direction of reinforcement is not important, only that reinforcement is performed. This hypothesis will be held to be confirmed if reinforcement is more common than non-reinforcement, and partially confirmed if there is an even or almost even blend between the two states.

8.7.1.2 Participants will rate the task as easier when they are being assisted by the framework

The initially fast setting (200ms) for registering a double-click combined with the oft-remarked problems older users have when working with double-clicking (c.f Jastrzembski, Charness, Holley, Feddon, 2005; Hawthorn, 2000; Hanson, 2001) imply that the task will be more difficult when the threshold is high than when it is low. As it is the intention of the plug-in to lower the threshold to a more comfortable value, the task should become easier as time goes by. Thus it is hypothesised that the participant's subjective rating of the task difficulty will be higher in the control session than it is in the experimental session.

8.7.1.3 Participants will complete the tasks quicker when they are being assisted by the framework

The framework will perform incremental corrections to the user's operating context on the basis of assessed difficulty, and so it is hypothesized that the time taken for the participant to complete the tasks will be lower as a result of the framework being active. The testing software maintains a log of time taken between successful double clicks as well as the number of successful clicks registered. In the experimental condition, it is hypothesized that time taken will be lower and number of successful clicks will be higher than in the control session.

8.7.1.4 Participants will find double-clicking a moving target to be too difficult in the control session

Double-clicking a moving target is not a common interaction requirement, and is chosen primarily to give the framework opportunities to make corrections. As such, it is hypothesized that the nature and unfamiliarity of the task will mean that abandonment rates for the moving target task will be high, if not 100%.

8.7.1.5 Participants will find double-clicking a moving target to be difficult but possible in the experimental session

While it is not anticipated that participants will find double-clicking a moving target to be easy when the framework is active, it is hypothesized that the corrective actions of the framework will make the task achievable, and thus there will be a statistically significant improvement between the two experimental conditions.

8.8 Multiple Plug-in Study

The Multiple Plug-in Study for the experiment involves the tasks one to five, completed in order. First, these tasks are completed without the use of the corrective framework, and then with. For half of the participants, this ordering is reversed.

The basic procedure of the second study follows that of the first study. In this experimental situation five plug-ins (Missed Clicks, Pointer Size, Mouse Trails, Double Clicks and Double Back) are activated and the interactions of these plug-ins over the user's entire experience are assessed. These plug-ins are discussed in detail in chapter six.

8.8.1 Hypotheses

The hypotheses that emerge from this study, as relating to the framework specifically, are as follows:

8.8.1.1 There Will Be a Substantial Amount of Variation Between Users With Regards to Plug-Ins

As a result of the way in which the framework selects plug-ins for correction, and as a consequence of the implications of dynamic diversity, it is hypothesized that each individual will end up with a different set of plug-ins having made a different set of corrections. It is also hypothesized that some participants will not be corrected at all. This would be consistent with the design of the framework, which is that different people will encounter different issues, and that different people will have different preferences for the corrections performed. It is expected that this will manifest itself in large standard deviations in both the number and type of reinforcements made by participants, and the number of times in which plug-ins presented themselves for reinforcement.

8.8.1.2 Each User Will Reinforce Plug-ins differently

It is not expected or desired that all plug-ins are equally useful to all participants, and so it is expected that plug-ins will be differently reinforced by different groups of participants. The framework maintains a roulette wheel of plug-in weightings, and this wheel will be altered as reinforcements are made by the participant. At the end of the process, there should be marked differences between the wheels as provided, and this will be reflected in high degrees of deviation between number of reinforcements and amount and direction of reinforcement.

8.8.1.3 Users will positively reinforce more than they positively punish

If the framework is consistently making corrections that are not desired, then it is not achieving its design goal of providing a realistic alternative to existing configuration regimes. This would be represented in the system as positive punishments being predominant over positive reinforcements. It is expected though that the generally beneficial nature of these corrections will result in the participants expressing approval on the whole. This would be represented in the system by positive reinforcements outnumbering positive punishments.

8.8.1.4 Users Will Reinforce Plug-ins with Invisible Behaviour

Both the double-click and the missed clicks plug-in in the system have relatively hidden implications for users. The impact of changing a double-click threshold will not be immediately observable, although a large number of corrections would be. Similarly, the impact of pointer precision can be subtle, and many users will not consciously register a change in their mouse behaviour.

However, each plug-in comes with a simple description of the changes that have been made, and it is hypothesized that this information will be sufficient in allowing users to make an informed decision. There exists only one plug-in for which the correction may be too subtle for users to appreciate, and that is the double-click plug-in. All the others have corrections that are sufficiently observable that participants will be able to see the impact directly. This hypothesis then requires that the amount of reinforcement applied to the double-click plug-in will be equivalent to the reinforcement performed on the others.

8.8.1.5 Participants will rate the tasks as easier when they are being assisted by the framework

Both the double-click and the missed clicks plug-in in the system have relatively hidden implications for users. The impact of changing a double-click threshold will not be immediately observable, although a large number of corrections would be. Similarly, the impact of pointer precision can be subtle, and many users will not consciously register a change in their mouse behaviour.

However, each plug-in comes with a simple description of the changes that have been made, and it is hypothesized that this information will be sufficient in allowing users to make an informed decision. There exists only one plug-in for which the correction may be too subtle for users to appreciate, and that is the double-click plug-in. All the others have corrections that are sufficiently observable that participants will be able to see the impact directly. This hypothesis then requires that the amount of reinforcement applied to the double-click plug-in will be equivalent to the reinforcement performed on the others.

8.8.1.6 Participants will complete the tasks quicker when they are being assisted by the framework

As is discussed in the hypothesis above, it is not expected that this is a hypothesis that will hold true for all tasks and all participants, and will instead be dependent on the plug-ins that were selected to perform corrections. However, it is expected that improvement will be recorded in all objective categories across all tasks.

8.9 Final Notes on this Study

The study outlined here is objectively quantitative (the time and correctness values logged by the software), and subjectively quantitative (measures of participant satisfaction). Each of these measure is aimed at analysing a different element of the framework – the objectively quantitative data permits a degree of certainty as to how objectively useful the framework is, but this by itself has little value if it is not also supported by the subjective perceptions of participants. Additionally, the nature of some of the framework suggest that it is possible that even in cases where there is no objective improvement that the framework simply makes for a more pleasant experience. This would be one possible interpretation of a case in which subjective improvement is recorded without corresponding objective improvement. The worst case scenario is that neither objective nor subjective improvement is recorded, which suggest that the design of the framework is fundamentally flawed. In order for a tool of this type to be useful, it must work and also be perceived to work.

8.10 Conclusion

The two studies outlined above draw together 80 hours of participant feedback in both subjective and objective form. It involves a number of different elements, but the core focus is on reinforcement learning within the framework. Thus, for the purposes of this study, it is not key that particular plug-ins are selected uniformly, or that plug-ins can diagnose user interaction issues with perfect precision. That software is capable of identifying user issues is not core to the study (and has already been addressed in studies such as Hurst, Hudson, Mankoff and Trewin, 2008). The novelty of this framework from the perspective of a user comes in the reinforcement learning mechanic itself. It is the expectation of this thesis that the framework can serve as an effective conduit between users and developers, and also as a suitable mechanism for deploying multiple accessibility solutions within a single application.

The first of these studies relates to whether reinforcement learning works for a single plug-in – whether the lack of granularity in user feedback is compensated by an aggregate series of individually binary data points. The second of the studies relates to whether it is an effective mechanism for deploying many plug-ins within a single context and ensuring that they all behave appropriately together – a successful mechanism will couple objective and subjective improvements in performance with high levels of support in terms of appropriateness, usefulness and tractability. The results for both of these studies are discussed in chapter ten. The questionnaire feedback received before and after the studies will provide a measure of understanding of both the confidence and competencies of the participants, and their reflective judgement on elements of the framework that are not amenable to information gathering during the tasks. The questionnaires completed before the experiment are discussed in the next chapter, while those that were completed after exposure to the framework are discussed in chapter ten.

9 A Study of Reinforcement Learning in Access – Participant Profiles

9.1 Introduction

Before the participants were exposed to the computer-based performance tasks, each was asked to fill out a questionnaire on their level of familiarity and comfort with regards to computer configuration. This information was then used to profile the individuals in the study and to show that the issues identified by Trewin (2000) were relevant to this group of individuals. As identified by Czaja and Lee (2007), every new study done on the generational issues related to computer use shows improvement amongst the older demographics, although a pronounced 'digital divide' still exists. A key justification for the design of the framework is that it bridges a gap between the confidence and capabilities possessed by older users, and the confidence and capabilities required to make configuration changes to a computer system. This initial exploratory research was focused on assessing whether or not this key justification is supported with regards to a modern grouping of participants.

9.2 Questionnaire Method and Analysis

9.2.1 Method

Each individual in the study was first given a short briefing with regards to the overall design of the experiment and what they would be asked to do within the studies. Before they were exposed to the computer based tasks, they were asked to fill out a pre-task questionnaire. The questionnaire consisted of twenty-one questions, each on a five point Likert scale, where one meant 'strongly disagree' and five meant 'strongly agree'. Questions focused primarily on confidence and comfort with computer configuration settings.

9.2.2 Analysis

Results were analysed by finding both the mean of the submitted results and also the arithmetic mode, the latter of which is included for completeness (c.f. http://poincare.matf.bg.ac.rs/~kristina//topic-dane-likert.pdf). For high level analysis, agree and strongly agree are coded into a single 'agree' category, and likewise for disagree and strongly disagree.

Strong agreement for a statement is defined as when the agreement rate is greater or equal to fifty percent and the disagreement rate is less than or equal to twenty-five percent. For strong disagreement with a statement, the reverse is held to be true. A statement is held to be contentious when both the agreement and disagreement rates are greater than or equal to twenty five percent. For questions where none of these are true, the result will be held to be leaning towards agreement or leaning towards disagreement depending on which is most appropriate.

9.3 Pre-Study Questions

This section will discuss the statements presented to participants in the study, and discuss both the rationale behind the question before presenting distribution of results and discussion of how they relate to the overall thesis under examination. The overall results are shown in table 4.

Statement	**Agreement Rate**	**Disagreement Rate**
1. I consider myself to be a novice computer user	21.05%	34.21%
2. I feel that computers are straightforward to operate	23.68%	52.63%
3. I usually understand the link between my actions and the outcome when working with a computer.	31.58%	28.95%
4. I am often unsure of the meaning when confronted with computing jargon	63.16%	23.68%
5. I have a solid understanding of the range of options available for me with regards to configuring my computer	21.05%	57.89%
6. I am comfortable with experimenting with computer settings in order to achieve the outcome I desire	23.68%	47.37%
7. I have confidence that I can correctly determine the impact of changing computer settings in most situations.	18.42%	65.79%
8. I know that computers have a wide range of settings designed to make it easier for me to perform the tasks I need to perform	36.84%	18.42%
9. I understand what is meant by the term ‘control panel’	47.37%	31.58%
10. I am aware of how to change the behaviour of my computer’s mouse to make it more suitable for my own requirements.	31.58%	50.00%
11. I know how to change the settings within my most familiar computer programs.	31.58%	57.89%
12. I am aware that it is possible to change the speed at which my mouse cursor moves in response to my movements.	28.95%	52.63%
13. I am aware that it is possible to change the speed at which a double click is recognized by the computer	31.58%	65.79%
14. I am aware of what is meant by ‘ease of access’ with regards to computers.	10.53%	60.54%
15. If it was necessary for me to change the way my computer was	18.42%	76.32%

set up, I would know how to do that.		
16. I usually work with an application as it is presented to me, without attempting to change the way it is set up.	65.79%	15.79%
17. I am anxious when changing a program setting with which I am not familiar.	68.42%	15.79%
18. I understand the difference between application-specific settings and system wide settings.	7.89%	76.32%
19. I am aware of how to change the behaviour of the keyboard to make it more suitable for my own requirements.	15.79%	60.53%
20. I consider myself to have requirements when working with a computer that are not met by the computer.	21.05%	50.00%
21. I am aware of what is meant by 'accessibility' with regards to computers.	21.05%	47.35%

Table 4 - Collated Table of Questionnaire Results

Each of these statements will now be addressed in turn. For the provided graphs, responses are indicated by SD (Strongly Disagree), D (Disagree), N (Neutral), A (Agree) and S (Strongly Agree).

9.3.1 Statement One - I consider myself to be a novice computer user

While all of the participants for the study were drawn from the SiDE user pool (as is noted in the acknowledgements for this work), there was still a wide difference in levels of computing familiarity within the group. Participants ranged from complete novices to those who had already learned how to configure their computers as a result of their own particular accessibility needs. Only twenty percent agreed with this statement, with 34% disagreeing. The strength of agreement was stronger than the strength of disagreement, as can be seen on the bar chart below. By the criteria outlined above, this statement is held to be leaning towards disagreement.

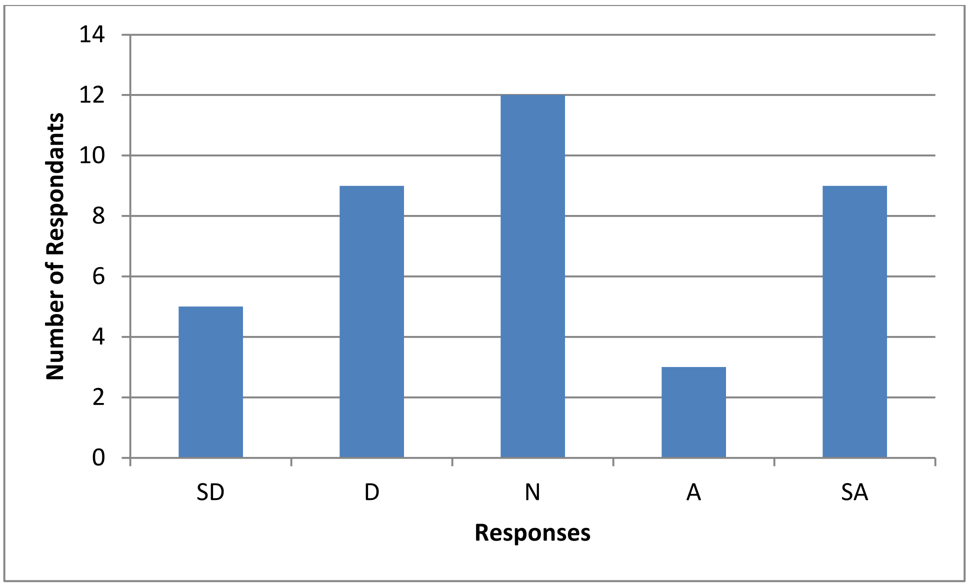


Figure 21 - Distribution of Results for Statement 1

It is not unexpected that there would be large differences in self-reported levels of experience with computers, however, it was hypothesized that novices would be the largest group within the study, whereas the results show that more people disagreed with the statement than agreed. This does not have a direct impact on the core hypothesis of the framework, which is that it will make computer configuration tractable for older, novice users. However, one potential conclusion of these results is that the size of the intended audience may be smaller than first expected. For an older user to be willing to make use of such software, they have to first see the need for it. Later questions specialize this inquiry into the elements of computer configuration that are handled by the framework, but if pitched as a tool for novice users, the results for this question strongly imply that a substantial number of older users will not see the tool as being aimed at them. However, an alternate explanation lies in the fact that individuals must register their interest in SiDE before they are contacted for studies, and thus the user pool may contain a disproportionate number of individuals with computing experience.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
21.05%	34.21%	44.74%	N	Agree	Leaning towards disagreement	1.17

Table 5 - Pre-study Statements 1

9.3.2 Statement Two - I feel that computers are straightforward to operate

It is commonly reported in the literature (c.f Newell et al, 2006; Dickinson et al, 2005; Czaja, 2005) that older users find computers to be less than straightforward to operate. However, the individuals drawn from SiDE self-reported a wide range of comfort levels, from those who had only just started using computers, to the other extreme where one participant was building ‘water cooled over-clocked double core systems’. An important aspect of the justification for this tool is that it fills a need for older users in providing a mechanism by which the accessibility configuration of a system can be made more straightforward by inverting the responsibility. It was hypothesized then that on the whole, the participants in the study would report low levels of agreement with this statement. This is confirmed by the data gathered, as is summarized in table 6.

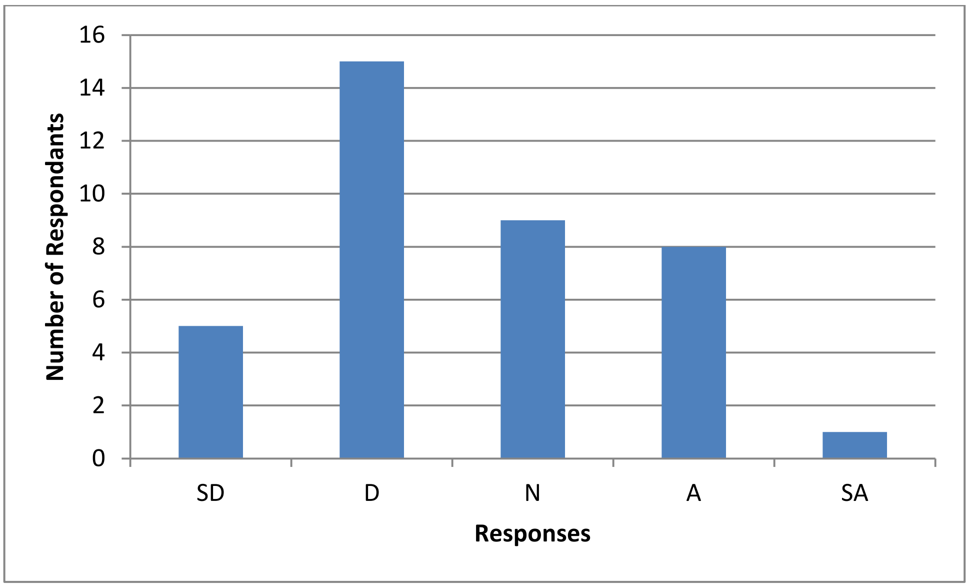


Figure 22 - Distribution of Results for Statement 2

The strong disagreement reported with this statement serves as evidence for the central hypotheses of this dissertation - despite the unexpected distribution of self-identified levels of experience, the majority of participants felt that computers were not straightforward to operate. The intention of ACCESS is that it turns complex configuration into a straightforward 'yes' or 'no', and while this statement does not directly reference computer configuration, it follows that if computers as a whole are not straightforward, then system configuration too is likely not straightforward. This is specialised in later statements.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
23.68%	52.63%	23.68%	D	Disagree	Disagree	1.07

Table 6 - Pre-study Statements 2

9.3.3 Statement Three - I usually understand the link between my actions and the outcome when working with a computer

The link between actions and outcomes within a computer system are often obscured by layers of abstraction, and as such the direct correlation between a user’s interaction and the consequences of that interaction are not necessarily obvious. This has implications for computer configuration, with regards to settings (such as mouse precision) which do not have clearly defined outputs, and where the lack of instant visual feedback, or worse, instant visual feedback that does not relate to actual changes (as an example of this, mousing over a style in Microsoft Word 2007) can make individuals less confident about their ability to configure their computer both in applications and in the operating system generally. It was anticipated that those layers of abstraction between action and outcome would result in individuals disagreeing with this statement, but the actual result is contentious when taken over the entire group:

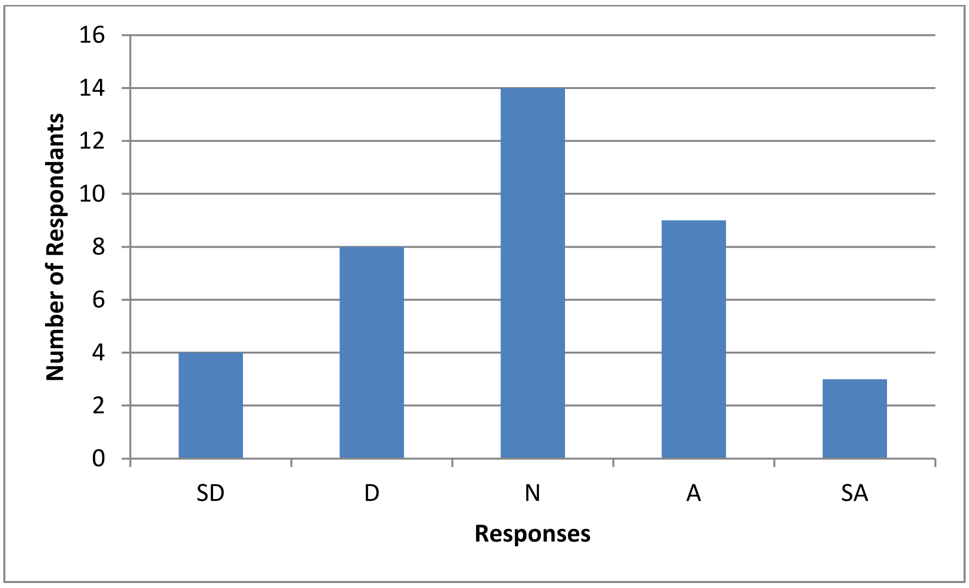


Figure 23 - Distribution of Results for Statement 3

Given the unexpected distribution of self-identified novices and non-novices, it is perhaps unsurprising that agreement with this statement would be more contentious than expected. This has implications for the issues of confidence discussed earlier in the framework justification - part of that hypothesized lack of confidence is down to a disconnect between actions and outcomes, and this is not well supported by these results.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
31.58%	28.95%	39.47%	N	Disagree	Contentious	0.98

Table 7 - Pre-study Statements 3

9.3.4 Statement Four - I am often unsure of the meaning when confronted with computing jargon

In discussions and studies, a common refrain of novice and older users is that computing jargon makes computer systems disproportionately difficult to operate (c.f. Newell et al, 2006; Goodman et al, 2003; Notess and Lorenzen-Huber, 2007). Within the configuration control panels, such jargon is often present and this has implications for how easily an individual can make changes to their system even when they are aware that such changes can be made. The language used almost unconsciously in computing interactions is replete with complex phrasing, seemingly interchangeable options, and undocumented choices. When changing the character repeat speed in a keyboard for example, two very similar sounding values must be altered (repeat delay and repeat rate). The first determines how long a key must be depressed before it begins to repeat, and the second determines how quickly it repeats. However, from discussions and debriefings with participants, it is clear that these two terms are not instantly tractable for novice users. Coupled to this is the fact that many options simply aren't documented at all in any way that is easily accessible, such as 'enhance pointer precision'. This is an option that is not immediately simple to explain and is provided without context or explanation in the mouse control panel on Windows systems. Searching for a simple explanation turned up the following via the internet:

> *What does Enhance Pointer Precision do? It's a simple concept. When enabled, the pointer moves more precisely when you move the mouse slowly, and more nimbly when you move the mouse quickly. It decouples pointer movement ever so slightly from a basic 1:1 relationship with mouse movement, and introduces something called the mouse acceleration curve.*
>
> *The translation from physical mouse movement to pointer movement is more sophisticated and more subtle than you might think. It's all documented in an excellent Microsoft article on mouse ballistics. It introduced me to the amusing concept of the mickey: the smallest unit of measurement that the mouse's hardware can produce.*
>
> (http://www.codinghorror.com/blog/2007/10/mouse-ballistics.html)

Not only are some options not effectively documented for an end-user, sometimes they are inadequately documented for developers too:

> *It is possible to edit these curves via the SmoothMouseXCurve and SmoothMouseYCurve registry settings, but there's absolutely no documentation I could find on these settings, so be careful. Getting the curve right is crucial.*
>
> (http://www.codinghorror.com/blog/2007/10/mouse-ballistics.html)

It was expected then that agreement with this statement would be high, as is confirmed by participant responses.

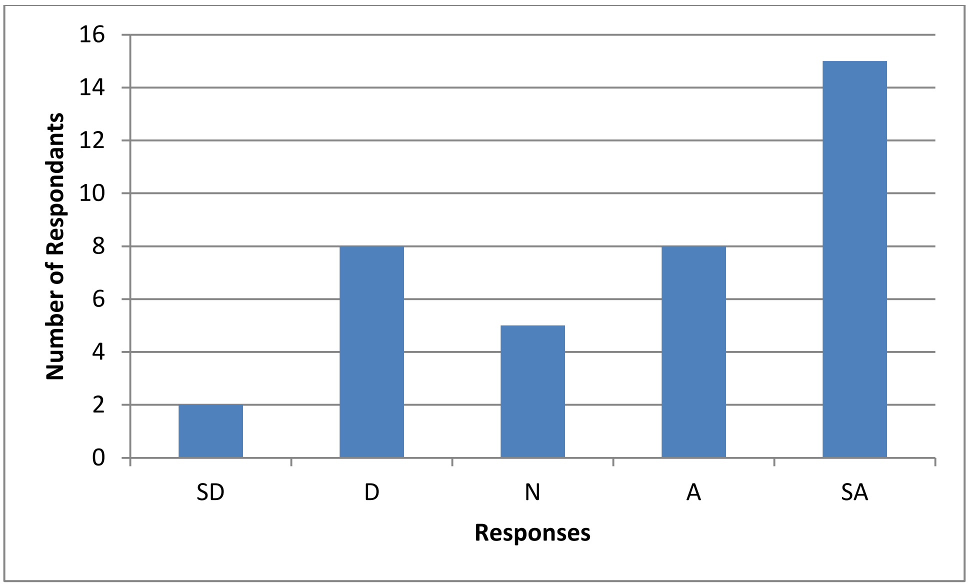


Figure 24 - Distribution of Results for Statement 4

This highlights an issue that is evident in the literature (c.f. Dickinson et al, 2005; Sayago and Blat, 2009; Goodman et al, 2003) - the choice of wording when providing software aimed at novices is a substantial issue. Plug-ins provide user feedback when they make a correction, and it is important that they describe what they have done in clear and simple language without reference to jargon or technical details.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
63.16%	23.68%	13.16%	SA	Agree	Agree	1.24

Table 8 - Pre-study Statements 4

9.3.5 Statement Five - I have a solid understanding of the range of options available for me with regards to configuring my computer

Most operating systems come complete with a large array of configuration options. An understanding of the range available is important in allowing individuals to make meaningful choices about their computer setups. However, a lack of context in the way these options are provided as well as the huge amount of trivial options presented equally prominently with the more critical options means that it is not simple to see what can be done and what is worth doing. As such, while users often have coping strategies for specific applications, they may be unaware of the ways in which correction can be done in a more consistent way across their entire system. In previous studies conducted for this dissertation, participants coped with suboptimal Word document visibility by increasing the font size of text or magnifying the document rather than choosing a more comfortable screen resolution. Disagreement with this statement was thus expected to be high, which was confirmed by participant responses.

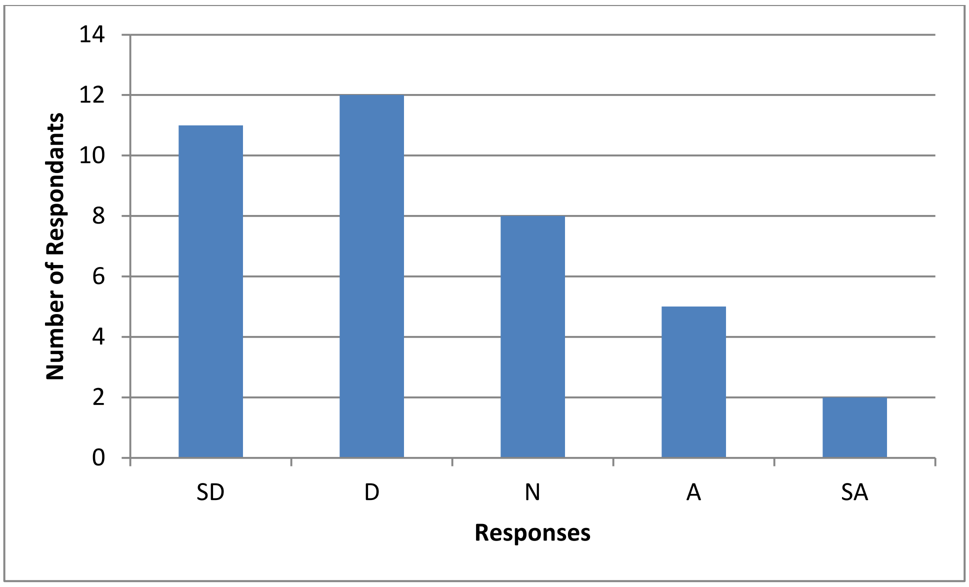


Figure 25 - Distribution of Results for Statement 5

The strong disagreement supports one of the framework justifications - that the intended audience for the tool do not have a strong understanding of what can be done to configure their computers. This is a general, self-identified perception, but is further specialised by later questions in the study.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
21.05%	57.89%	21.05%	D	Disagree	Disagree	1.25

Table 9 - Pre-study Statements 5

9.3.6 Statement Six - I am comfortable with experimenting with computer settings in order to achieve the outcome I desire

While documentation exists on almost every operating system setting that may be available to a user, it is often locked away in obscure developer references and not usually discussed in introductory computer books or computer classes. In such circumstances, it is likely the level of comfort an individual has with experimentation is an indicator of the likelihood they will be able to find the system settings they need in order to properly configure their systems. Dickinson et al (2002) discuss this in relation to 'learning while doing'.

However, the elements of computer configuration that have been discussed in previous chapters often discourage older and novice users from experimenting with the system. Users often feel anxious when changing settings with which they are not already familiar (see later in this chapter), and this can have an impact on willingness to explore configuration options within a system (c.f. Wright et al, 2003; Hawthorn, 2005). Disagreement with this statement was expected to be high, but the participant feedback was instead more nuanced – the feedback leans heavily towards disagree, but does not meet the criteria outlined above required to interpret the results as a clear disagreement.

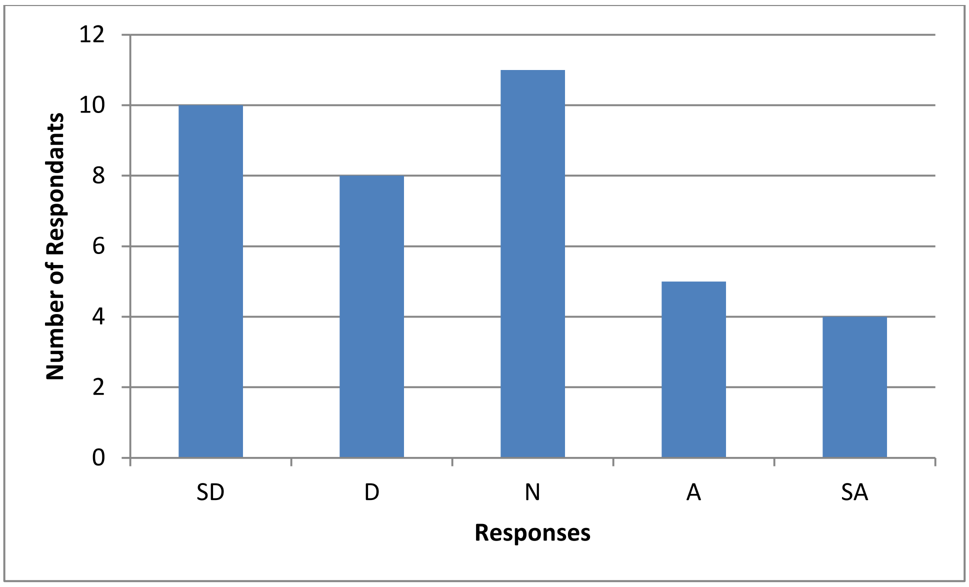


Figure 26 - Distribution of Results for Statement 6

The design of the ACCESS system is such that it negates the need for experimentation (at least in those circumstances where plug-ins are available to compensate). Participant debriefings however indicated that the majority of participants felt that their computer was a tool, rather than something they should be required to experiment with in order to make its performance optimal. Computers were seen as a means to an end, rather than something individuals would use for its own sake. Thus, even though the feedback here is more nuanced, one possible explanation is that that individuals would prefer not to have to experiment, even if they were comfortable doing so, was observed during the study and follow-up focus group.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
23.68%	47.37%	28.95%	N	Disagree	Leaning to Disagree	1.27

Table 10 - Pre-study Statements 6

9.3.7 Statement Seven - I have confidence that I can correctly determine the impact of changing computer settings in most situations

In statement three (I usually understand the link between my actions and the outcome when working with a computer) participants were asked if they usually understood the link between their actions and their outcomes. Statement seven provides a solid context for that statement by linking it to the specifics of computer settings. As with statement three (I usually understand the link between my actions and the outcome when working with a computer), disagreement with this statement was expected to be high, and this is confirmed by the participant responses. This is at odds though with the response to statement three which is held to be contentious. It is hypothesized that this is due to the open nature of statement three which does not focus the participant on their computer configuration behaviour. This suggests that individuals feel as if they understand the impact of their actions when working generally with a computer, but feel that assessing the impact of changing computer settings is a substantially different activity.

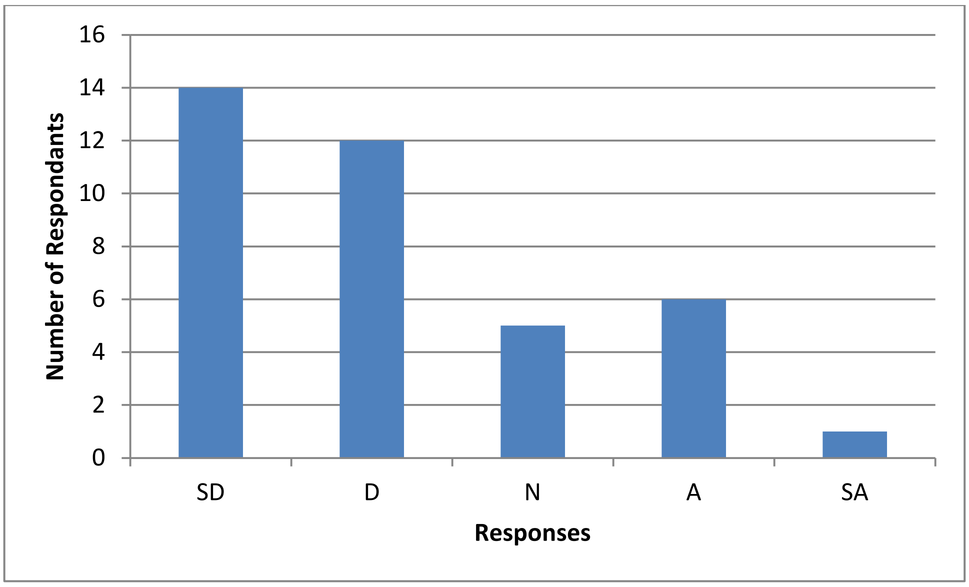


Figure 27 - Distribution of Results for Statement 7

The confidence levels for the statement 'I usually understand the link between my actions and the outcome when working with a computer' were evenly distributed. This statement though focuses on aspects germane to the framework, and the associated disagreement rate is much more pronounced. Not understanding the link between changing a setting and its impact heavily implies that users will not have confidence when changing their computer settings, and this is consistent with the design of ACCESS.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
18.42%	65.79%	15.79%	SD	Disagree	Disagree	1.21

Table 11 - Pre-study Statements 7

9.3.8 Statement Eight - I know that computers have a wide range of settings designed to make it easier for me to perform the tasks I need to perform

As with the statement above, this statement seeks to provide a concrete context for the more general case provided in statement five (I have a solid understanding of the range of options available for me with regards to configuring my computer). It brings the focus directly to computer configuration. The more general statement assesses the level of familiarity participants have with computer settings generally, while this statement seeks to assess the level of familiarity with the settings that are most important on a day to day basis.

As with statement five, disagreement with this statement was expected to be high, but the participant feedback is instead more nuanced suggesting that while most users do not feel as if they have a solid understanding of the full range of options, more believe that they have an understanding of the options that actually matter with regards to performing their day to day tasks.

The profile of responses though shows a very strong trend towards neutral, showing that on the whole people are unsure rather than comfortable with what they can do to make their computer easier to use.

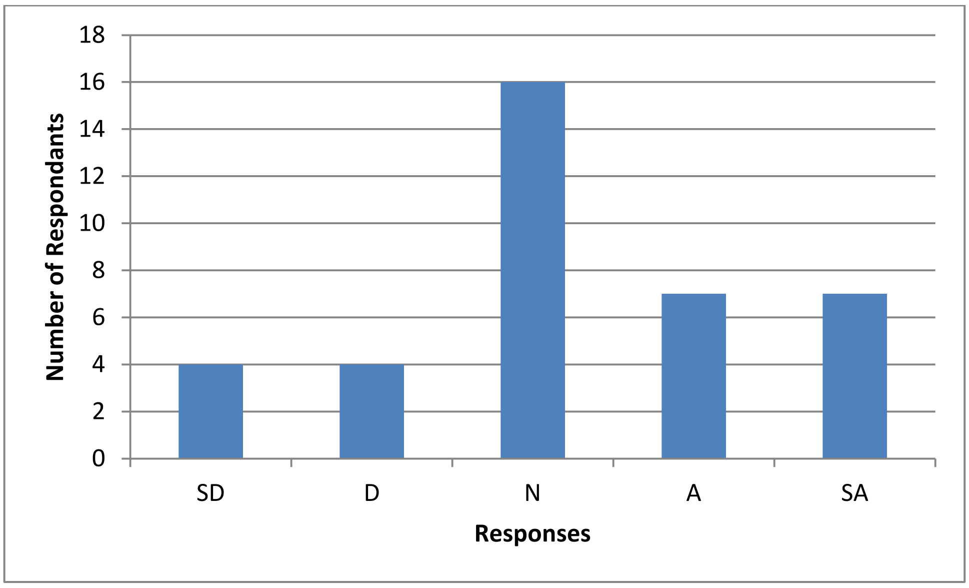


Figure 28 - Distribution of Results for Statement 8

Disagreement with this statement was anticipated, which would support the hypothesis with regards to older users lacking in the knowledge of what can be changed. This was not observed in the gathered data.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
36.84%	18.42%	44.74%	N	Disagree	Leaning Towards Agree	1.22

Table 12 - Pre-study Statements 8

9.3.9 Statement Nine - I understand what is meant by the term control panel

Familiarity with the control panel defines how likely an individual is to be able to reconfigure the settings of their system to meet their own requirements. This familiarity begins with an understanding of what the term means. As with the best computing metaphors, users should be able to leverage their understanding of the term in its general sense to understand its purpose within a computer system. However, while many users know that there is such a thing as the control panel (because it usually appears quite prominently on the start menu), it is less clear as to how many understand what is meant by the term. Disagreement with this statement was expected to be high, but the result is instead contentious although it leans slightly towards agreement.

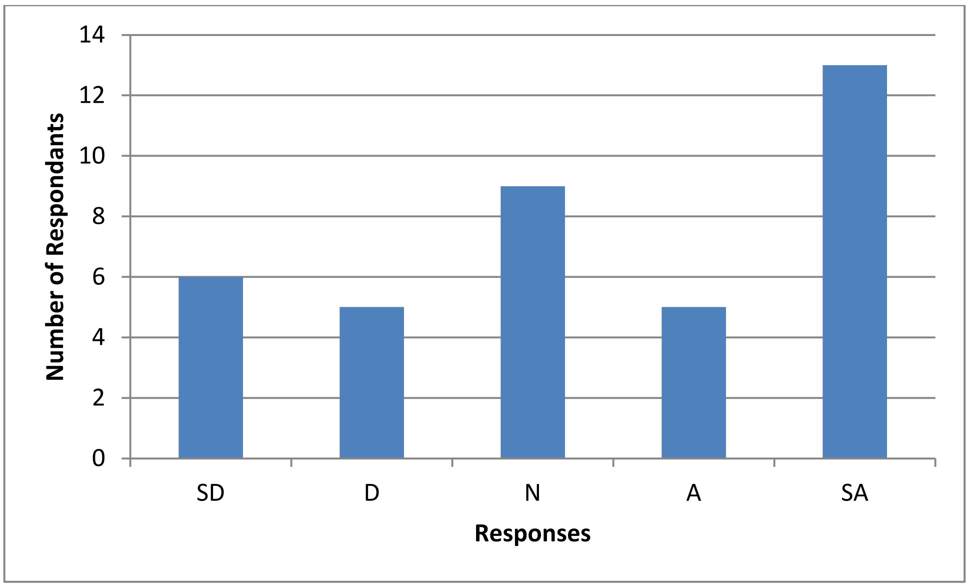


Figure 29 - Distribution of Results for Statement 9

It was anticipated that the control panel would be an unfamiliar metaphor for the majority of the participants in the study. However, the results instead show that the situation is more contentious - a sizable minority disagreed with the statement, but it would appear that on the whole participants are at least aware of its existence and function in the general sense.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
47.37%	31.58%	21.05%	SA	Disagree	Contentious	1.60

Table 13 - Pre-study Statements 9

9.3.10 Statement Ten - I am aware how to change the behaviour of my computer's mouse to make it more suitable for my own requirements

The tasks in this study were focused primarily on mouse interaction. As such the level of familiarity individuals have with regards to such configuration is an important aspect in assessing the value of a tool of this nature. Later on in the study, participants were asked for their level of comfort in changing the system to achieve specific goals, but this general statement serves to further specialize statement eight (I know that computers have a wide range of settings designed to make it easier for me to perform the tasks I need to perform).

Disagreement with this statement was expected to be high, but the result is instead contentious although leaning very substantially towards disagreement.

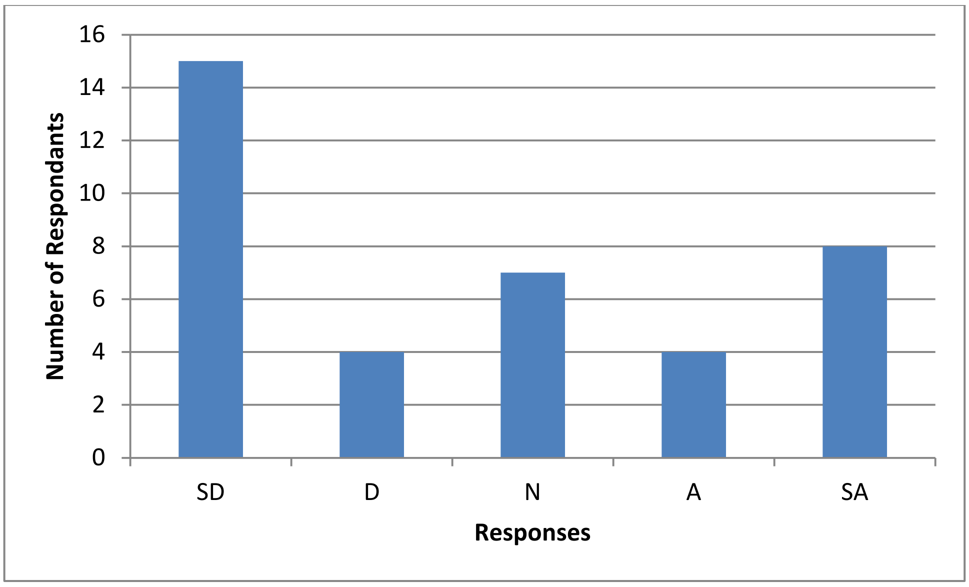


Figure 30 - Distribution of Results for Statement 10

This serves as some support for the hypothesis regarding the knowledge users have of how to make changes, but the relatively high agreement rate shows that a substantial minority of participants feel that they already know how to make changes to the way the mouse works. Thus a tool that automatically assesses and corrects problems would be perceived to be less useful for this group.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
31.58%	50.00%	18.42%	SD	Disagree	Contentious	1.62

Table 14 - Pre-study Statements 10

9.3.11 Statement Eleven - I know how to change the settings within my most familiar computer programs

Individuals often have particular preferences with regards to certain application configurations. Common examples include levels of magnification in word processors or size of buttons in internet browsers. However, these settings are seldom honoured between different applications, and such configuration does not necessarily lead to a more generally satisfying user experience outside of that specific context. Individuals changing font sizes rather than screen resolution are an example of this in practise. It solves the problem in the short term for a single application, but causes problems for example when the document is printed. Many individuals have coping strategies for their most used applications, and thus it is expected that participants in this study would express agreement with this statement. However, participant feedback marks the statement instead as contentious, and leaning towards disagreement.

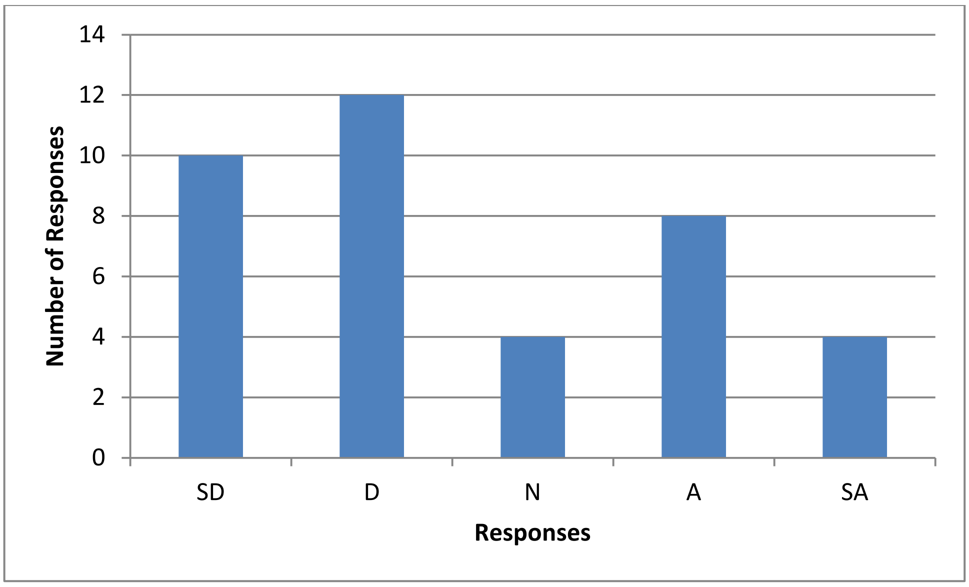


Figure 31 - Distribution of Results for Statement 11

This was an unexpected result. Much of the justification of the framework has been predicated on corrections at a system level rather than an application level. While this doesn't invalidate that as a justification, it does strongly imply that a wider focus than is currently permitted by the framework would be a worthwhile avenue of investigation.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
31.58%	57.89%	10.53	D	Agree	Contentious	1.31

Table 15 - Pre-study Statements 11

9.3.12 Statement Twelve - I am aware that it is possible to change the speed at which my mouse cursor moves in response to my movements

The speed at which the mouse cursor moves can have a considerable impact on the usability of a computer system (Hawthorn, 2000; Jastrzembski et al, 2005), and windows systems in particular come with two main ways in which the speed of the mouse cursor can be altered – either through the pointer speed, or the pointer precision. The computer that individuals used for the computer based trials was set up to have the cursor speed at the extremes of what can reasonably be configured. This statement further specializes statement ten (I am aware how to change the behaviour of my computer's mouse to make it more suitable for my own requirements) in order to relate it specifically to the most straight-forward of the exercises that individuals perform during the computer based tasks. A core assumption of the framework is that it will be possible to provide corrective action based on analysed input data, and that corrective action will be able to compensate for reduced user confidence or knowledge of configuration options. As such its feasibility requires a user-base who are either unaware that the options exist, or unaware how to make the changes, or lacking in confidence in assessing the impact of the changes. Disagreement with this statement was expected to be high, but the result is instead contentious leaning strongly towards disagreement:

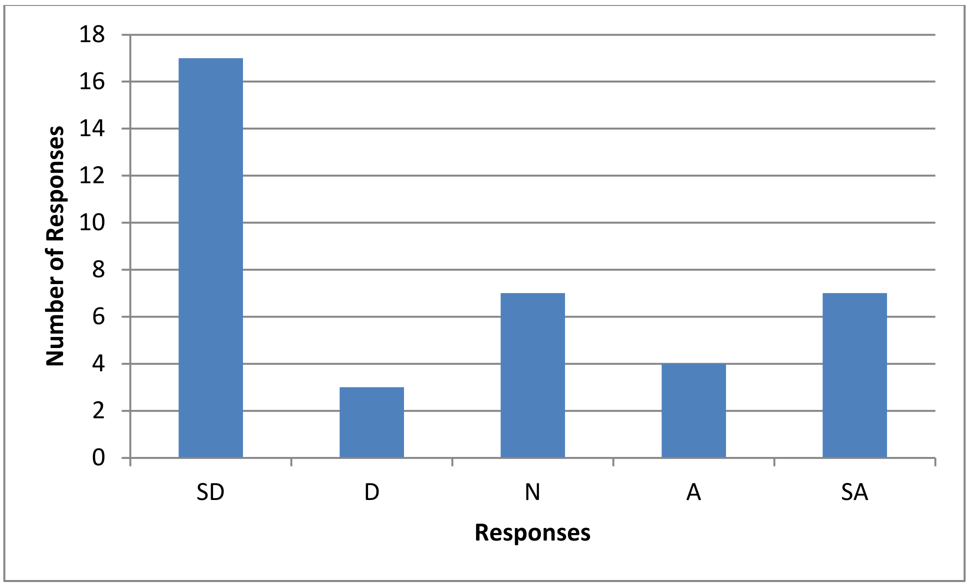


Figure 32 - Distribution of Results for Statement 12

This is one of the most visible impacts of computer configuration, as the mouse cursor is an interface element common to most operating systems and contexts. It is perhaps then unsurprising that such a large minority would be aware that it is one of the things that can be configured. However, even given the contentious results, more than fifty percent of participants do not feel confident in agreeing with the statement. This is a greater proportion than would be implied from the self-reported answers to statement one (I consider myself to be a novice computer user) regarding identification of experience. Despite the contentious nature of the results, it does provide support for the hypothesis that users would benefit, in the majority of cases, from a tool that addresses this consideration.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
28.95%	52.63%	18.42%	SD	Disagree	Contentious	1.59

Table 16 - Pre-study Statements 12

9.3.13 Statement Thirteen - I am aware that it is possible to change the speed at which a double click is recognized by the computer

As with statement twelve (I am aware that it is possible to change the speed at which my mouse cursor moves in response to my movements), this statement specializes an earlier statement (in this case, ten) in reference to specific tasks within the computer portion of the study. Double-clicking is widely acknowledged as a troublesome technique for older users (c.f Jastrzembski et al, 2005; Hawthorn, 2000; Hanson, 2001), and one of the plug-ins in this study addresses this concern. However, as with statement twelve (I am aware that it is possible to change the speed at which my mouse cursor moves in response to my movements), the assumption at the core of this framework is that it can make changes that users cannot or will not make for themselves. This assumes that there is a substantial proportion of older users for which these kind of corrections would be useful. Disagreement for this statement was expected to be high, but as with statement twelve the results are instead contentious, leaning strongly towards disagreement.

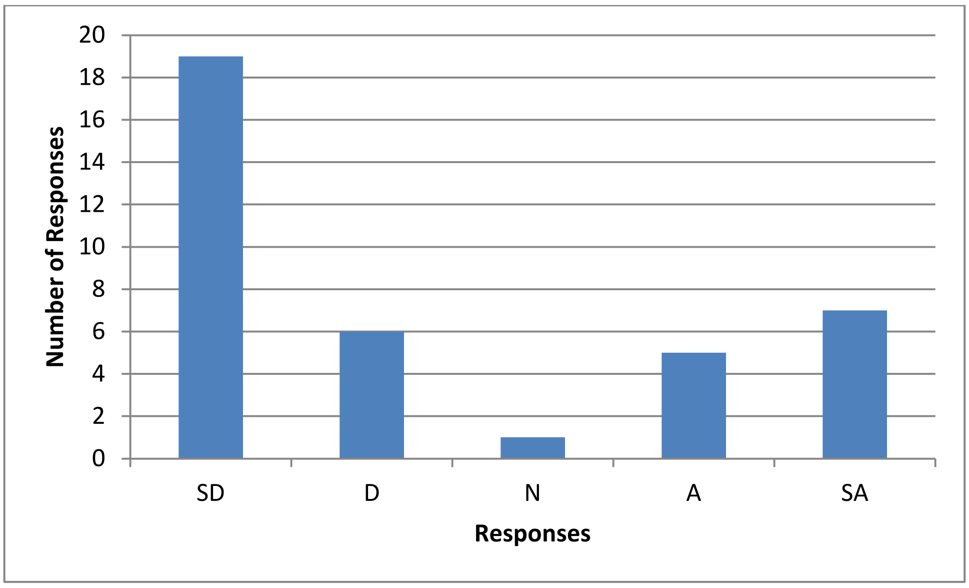


Figure 33 - Distribution of Results for Statement 13

Only one participant was unsure on this score - all other participants expressed a preference, and more than twice as many disagreed as agreed. Thus, while the results are contentious, it provides additional support for the hypothesis that users often do not know what can be changed within their systems, and thus provides additional evidence that the framework is aimed at addressing a need that actually exists.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
31.58%	65.79%	2.63%	SD	Disagree	Contentious	1.67

Table 17 - Pre-study Statements 13

9.3.14 Statement Fourteen - I am aware of what is meant by 'ease of access' with regards to computers

The terms 'accessibility' and 'ease of access' are used regularly by those who are familiar with the issues of computing usability. Perhaps though these terms are not useful for computer novices because the selection of such words one suspects is sometimes as much a matter of corporate branding as it is of clarity in expression. On Windows XP systems, there is a control panel marked 'accessibility', and on Windows Seven this control panel is called 'ease of access'. On OS X it is called 'Universal access'. All three terms relate to the same basic partitioning strategy of grouping these substantial system modifications into their own control panel. It was expected that agreement with this statement would be high, indicating that this term is tractable for older computer users, but instead it was a statement which had a high incidence of disagreement. This suggests that the use of such terms is counter-productive in computer configuration with older users. Older users as a group are also less likely to self identify as having accessibility issues due to the slow decline associated with aging (Gregor et al, 2002), and so a control panel that may contain useful corrections is considered to be both unclear in meaning and of limited relevance.

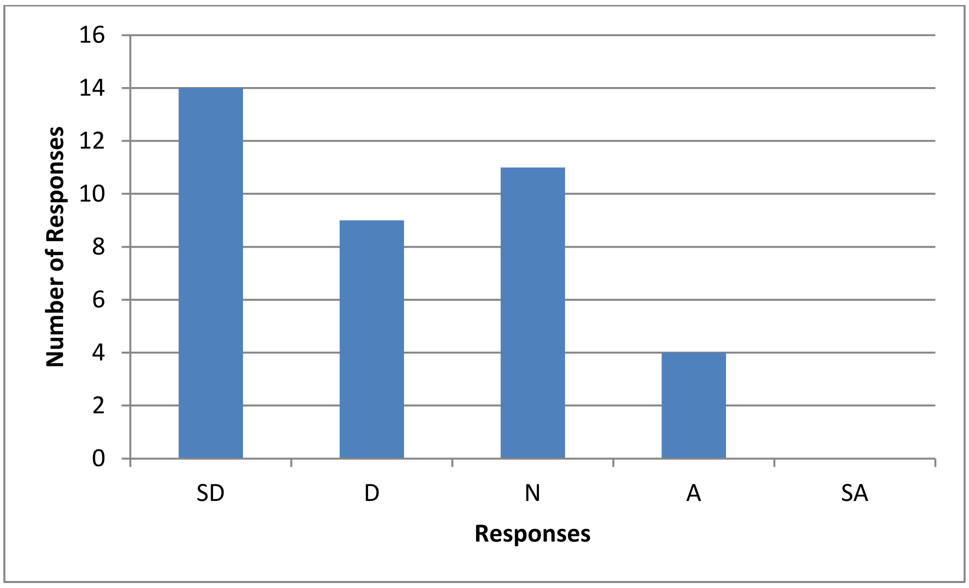


Figure 34 - Distribution of Results for Statement 14

The results of this statement are illuminating, but do not directly impact on the design or the justification of the tool. They do suggest however that even in cases where users are familiar with certain control panels, they will not appreciate the potential benefits to be found within control panels explicitly designed to provide simple access to accessibility support.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
10.53%	60.53%	28.95%	SD	Agree	Disagree	1.03

Table 18 - Pre-study Statements 14

9.3.15 Statement Fifteen - If it was necessary for me to change the way my computer was set up, I would know how to do that

Familiarity with the range of accessibility options present gives one view of individual comfort with regards to an individual's ability to configure their computer. Actually converting that familiarity into the knowledge of how that is done represents a second barrier to configuration. This statement attempts to assess the general comfort participants had with turning their abstract knowledge of what can be done into what they know how to do. Disagreement with this statement was expected to be high, as was confirmed by the participant responses. However, in order to avoid slanting the statement towards specifics, the statement is more of general diagnostic use than relating to any specific situations. It represents the general feeling of comfort when reconfiguring a system only.

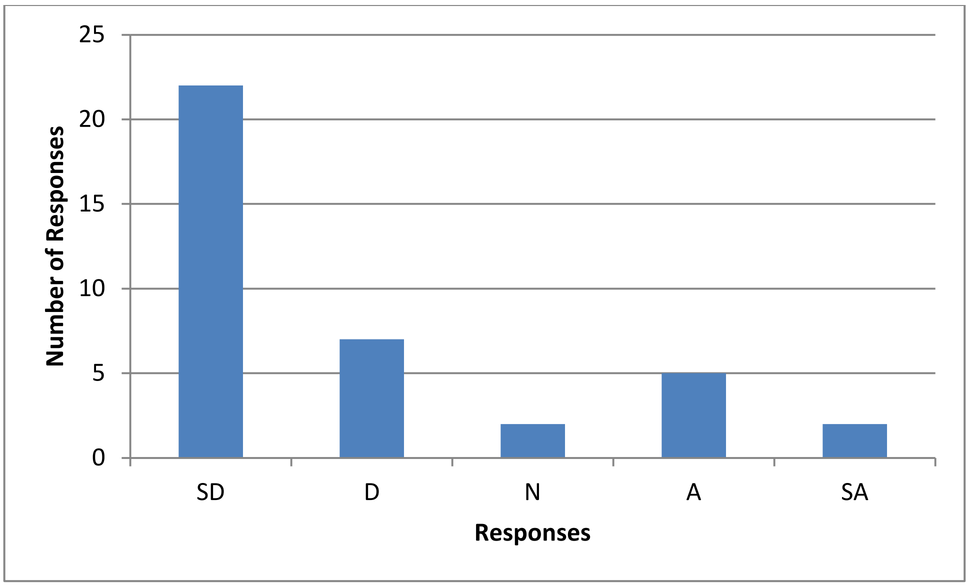


Figure 35 - Distribution of Results for Statement 15

This lack of solid context means that the statement can only be taken as suggestive of a knowledge deficit with regards to computer configuration. As such, its impact on the justifications for the design of the framework is minimal - the results are strongly suggestive that, as a general measure, participants do not feel as if they are capable of reconfiguring their computers.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
18.42%	76.32%	5.26%	SD	Disagree	Disagree	1.35

Table 19 - Pre-study Statements 15

9.3.16 Statement Sixteen - I usually work with an application as it is presented to me, without attempting to change the way it is set up

When confronted with an unfriendly operating environment for an application, the electronic notes study discussed in chapter seven suggested that users will simply attempt to work with the environment as it is presented without attempting to make any adjustments. In such circumstances, it does not matter if an individual has both the knowledge and the confidence needed to make changes because the changes are not being made regardless. This has considerable implications for the design and deployment of systems and applications. A high agreement rate implies that the status quo will be accepted regardless of how suboptimal it may be for users, something which is a recognized psychological principle (this is known as the status quo bias - an interesting treatment of this cognitive bias can be found in Kahneman, Knetsch, Thale, (1991). Agreement with this statement was expected to be high, as is borne out by the participant responses:

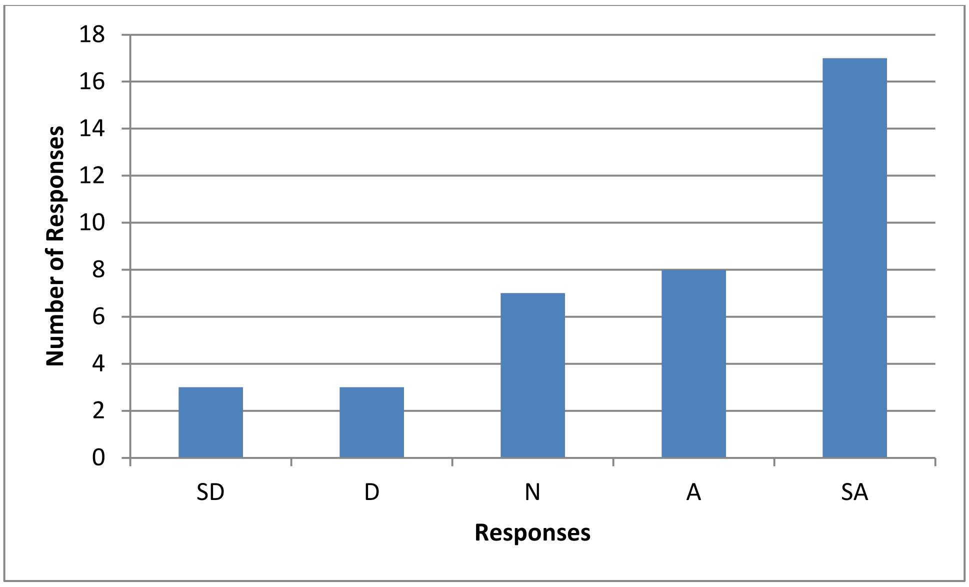


Figure 36 - Distribution of Results for Statement 16

This confirms the experimental observations discussed in chapter five with regards to windowed applications. The choice of a default is powerful, as is discussed in many books on psychology and influence management (c.f Cialdini, 2001; Thaler and Sunstein, 2008). The fact that participants choose to work with an application as it is presented has a number of implications for what default settings should be with a computer system. A typical guideline in the user interface literature is to 'design for the most common 80%', as expressed in the following extract:

> *During the design process, if you discover problems with your product design, you might consider applying the 80 percent solution—that is, designing your software to meet the needs of at least 80 percent of your users. This type of design typically favors simpler, more elegant approaches to problems.*
>
> (http://jurgenleckie.wordpress.com/tag/80-percent-solution/)

This particular piece of advice is drawn from Apple's Human Interface Guidelines (http://www.multimedialab.be/doc/tech/doc_osx_hi_guidelines.pdf) and even a casual search of Google will reveal the '80 percent solution' being referenced repeatedly in the context of developing user interfaces.

The power of the default value though means that not only does this disadvantage older users in the first instance, in most cases it will continue to disadvantage because no attempt will be made to readjust the system.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
65.79%	15.79%	18.42%	SA	Agree	Agree	1.23

Table 20 - Pre-study Statements 16

9.3.17 Statement Seventeen - I am anxious when changing a program setting with which I am not familiar

This statement is related to statement six (I am comfortable with experimenting with computer settings in order to achieve the outcome I desire) and assesses the level of anxiety individuals feel when changing settings with which they are not already familiar. As is discussed above, experimentation is an important aspect in understanding the full range of settings and options available in a computer system. Such experimentation though is discouraged when individuals feel anxious that they will be unable to undo changes they have made, or will simply not notice changes until it is too late. Older users in particular will tend to blame themselves rather than badly designed software in such circumstances (Eisma et al, 2004), which is likely to increase the level of anxiety an individual feels when experimenting. Agreement with this statement was expected to be high, as was confirmed by participant responses.

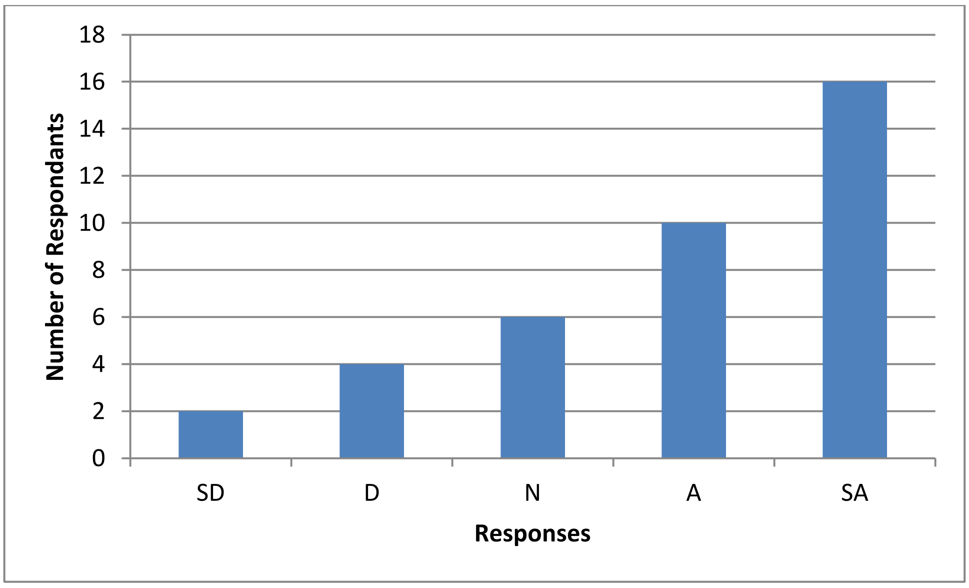


Figure 37 - Distribution of Results for Statement 17

Since much of the functionality provided by a computer is partitioned away in the menus of applications and the general system control panels, experimentation is the only realistic way to gain a knowledge of what is possible. While the internet remains a useful resource for investigation, its lack of context and structure means that it is not an effective tutorial, and only supplements directed individual experimentation. In many cases, operating systems and software do not come with paper manuals any more, preferring instead to provide documentation in an on disc PDF file. Computer textbooks and guides, due to space limitations, usually do not contain instruction regarding the more obscure functions available. In such an environment, anxiety in experimentation serves as a considerable barrier to exploring the options within a computer system. The high rate of agreement then serves a support for the framework justifications.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
68.42%	15.79%	15.79%	SA	Agree	Agree	1.11

Table 21 - Pre-study Statements 17

9.3.18 Statement Eighteen - I understand the difference between application-specific settings and system wide settings

Some older users, when presented with an application that is sub-optimally configured, will attempt to make corrections within the bounds of a single application. Since these corrections are rarely honoured across applications, the result is either a need to individually configure every application to meet a specific accessibility need, or frustration that some applications are set up appropriately and others are not. For a small pool of applications, this may prove an acceptable coping strategy. However when working with many applications or experimenting with new functionality, it remains a problem that can only be fixed by making the adjustments as system wide corrections rather than application specific. This statement assesses whether the distinction between changes that are made within applications and those that are made across a whole system is one that is understood by participants. Disagreement with this statement was expected to be high, as was confirmed by responses.

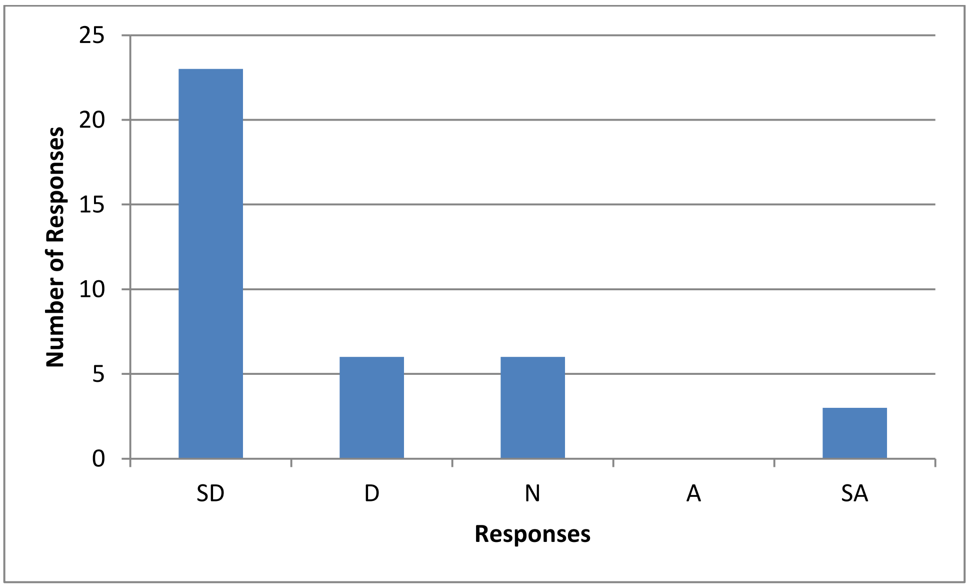


Figure 38 - Distribution of Results for Statement 18

Application-specific coping strategies are non-scalable and often come with problems of their own. Changing the font size in a document may make it more readable, but will have an impact on printing. In addition, it is a correction that must be performed every time a new document is encountered. Addressing accessibility concerns at the system level should be the first priority, as that ensures that the concerns are met for all applications and not just the one that has been specifically tailored. However, in making these system corrections, an individual must be aware of the difference - the responses to this statement show overwhelmingly that this is not a distinction appreciated by participants.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
7.89%	76.32%	15.79%	SD	Disagree	Disagree	1.31

Table 22 - Pre-study Statements 18

9.3.19 Statement Nineteen – I am aware of how to change the behaviour of my keyboard to make it more suitable for my own requirements

There were no specific keyboard tasks in this study, although several proof of concept keyboard plug-ins exist as exemplars for the framework. The expectation for this statement is that the profile of answers would be roughly similar to those of statement ten (I am aware how to change the behaviour of my computer's mouse to make it more suitable for my own requirements) which had a contentious profile. The actual results showed that this statement can be considered as 'disagree' suggesting that more participants were aware of the need for mouse configuration than keyboard configuration. The options available for keyboards are somewhat limited outside of the specific accessibility options, and the design of a keyboard permits for corrective activity to occur outside the context of the system such as angling it in a more comfortable way.

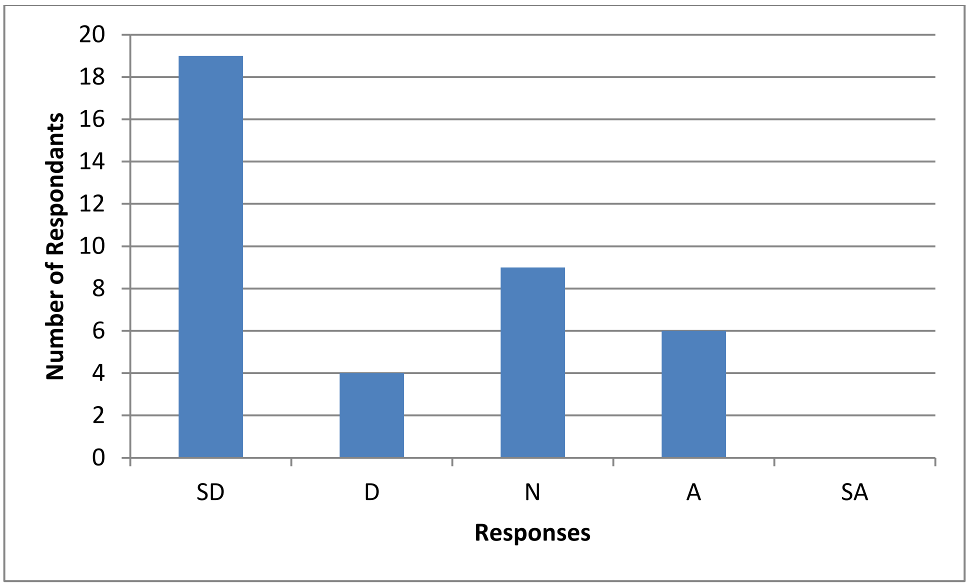


Figure 39 - Distribution of Results for Statement 19

Thus, while specific keyboard accessibility tools were not addressed in the context of this research, the results imply that such tools would be useful to a wider pool of people than the mouse corrections purely as a consequence of the fact more participants registered awareness of these options than they did of keyboard options.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
15.79%	60.53%	23.68%	SD	Disagree	Disagree	1.21

Table 23 - Pre-study Statements 19

9.3.20 Statement Twenty – I consider myself to have requirements when working with a computer that are not met by the computer

An important aspect of adjusting the configuration options of a computer system is identifying the need for changes. Older users in general will not self-identify themselves as having accessibility difficulties (Gregor et al, 2002), and will often report no substantial ease of access concerns. As such, it was expected that older users would not consider themselves to have special requirements that were not met when working with a computer system. Disagreement with this statement was expected to be high as was borne out by the participant responses.

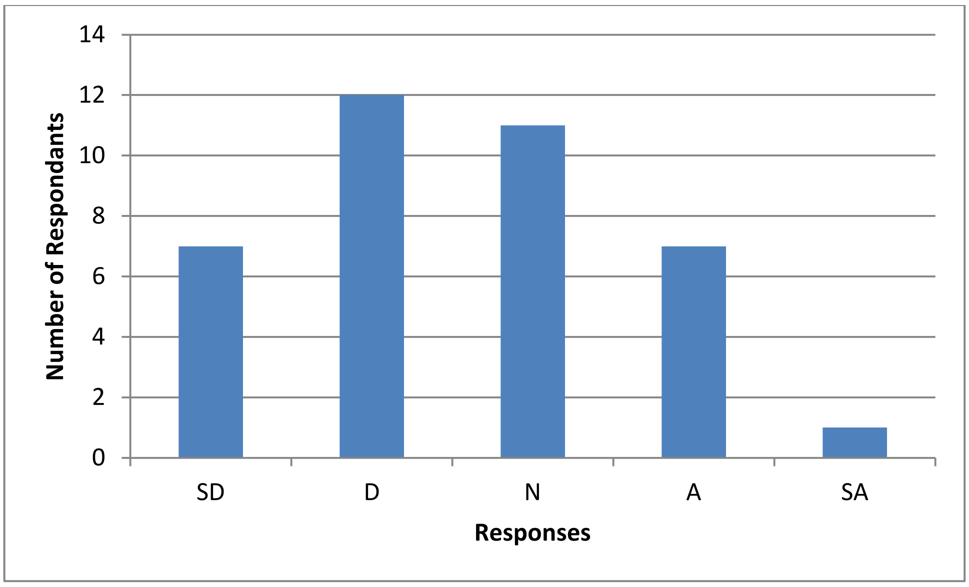


Figure 40 - Distribution of Results for Statement 20

During the testing of this framework, it identified an accessibility concern that I had that wasn't being met by my system. My comfortable double-click threshold was 100ms higher than my system was configured - this was a surprise to me, as I consider myself to have no accessibility concerns when working with my computer. While this is only an anecdotal data point, it does show that an individual's perception of where accessibility improvements can be made will differ from that which can be algorithmically detected. The fact individuals often do not feel they have any substantial accessibility concerns does not mean that they do not have a realistic self-appraisal. The inevitable consequences of aging though suggest that many will benefit from some tweaks to their system settings. Thus, the framework as designed would address this need by ensuring individuals who do not feel they require any adjustments have the need independently and quantitatively defined by an objective algorithm.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
21.05%	50.00%	28.95%	D	Disagree	Disagree	1.10

Table 24 - Pre-study Statements 20

9.3.21 Statement Twenty One – I am aware of what is meant by ‘accessibility’ with regards to computers

The phrase ‘ease of access’ as a control panel option was introduced with Windows Vista, and the clarity of that term was assessed in statement fourteen (I am aware of what is meant by ‘ease of access’ with regards to computers). ‘Accessibility’ is a more common term with a longer lineage, and thus it was expected that it would be more tractable for participants than the phrase ‘ease of access’. Agreement was thus expected to be high for this statement, and also higher than the agreement rate in statement fourteen. The feedback on this statement was more nuanced than expected, showing a strong leaning towards disagreement, but not enough to classify it as such according to the criteria outlined at the start of the chapter. However, the agreement rate was double that of the phrase 'ease of access', suggesting that the move to the new phrasing in later versions of Windows comes with a corresponding reduction in the clarity of the jargon. In both cases, the clarity of the term has low rates of agreement, suggesting that segregating such functionality away from its context is a counterproductive strategy.

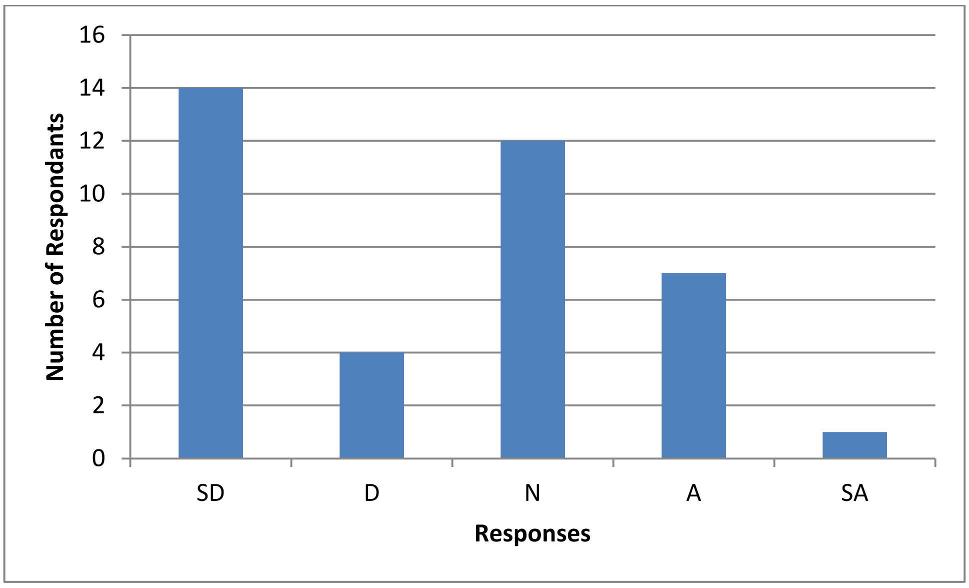


Figure 41 - Distribution of Results for Statement 21

The conclusion that can be drawn from this statement are those of statement fourteen - the impact on the justifications for the framework is minimal. It does show that the functionality partitioning done in most operating systems can be counterproductive, and that the language choice used for these kind of options must be careful considered for maximal tractability.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
21.05%	47.37%	31.58%	SD	Agree	Leaning towards Disagree	1.22

Table 25 - Pre-study Statements 21

9.4 Discussion

The results of this pre-task questionnaire demonstrate support for the fundamental assumptions of the framework. They also identify several issues with regards to the method by which such configuration options are made available to users. The strategy of segregating accessibility support away from its context in terms of mouse and keyboard options seems to suffer with regards to the intention of the related control panels (statements twenty-one and fourteen). In addition, the literature suggests that options that are visually identifiable as accommodating for an accessibility issue can be stigmatizing

(Sokoler and Svensson, 2007), and thus they would benefit from being located within the appropriate control panel as well as within their own specific panel. Participants did not feel that they had any specific requirements that were not met by the system (statement twenty - I consider myself to have requirements when working with a computer that are not met by the computer), which is consistent with the literature on 'healthy older users' (Gregor et al, 2002). Computers on the whole are not considered to be straightforward (as evidence by statement two - I feel that computers are straightforward to operate) by the participants in this study, again consistent with the literature on the topic of older users and computers.

It is clear from the data collected that the justification for the framework is not universally applicable. However, even in those statements where the results are contentious it is clear there is considerable support for the framework even when it cannot be universally assumed. That the framework would be of more use to novices than it would be to experts is uncontroversial, and the profile of the users in this study shows considerable support for the justifications made for the framework's design. The requirement for a framework like ACCESS, from a user's perspective, was predicated on three primary arguments, each of which will be discussed in relation to the collected data.

9.4.1 Older users do not know what can be changed

This argument is assessed in statements five, six, eight, nine, twelve and thirteen. The following firm conclusions can be drawn from the data gathered from participants:

- On the whole, participants do not have a firm understanding of the range of options they have available to them (statement five - I have a solid understanding of the range of options available for me with regards to configuring my computer)
- On the whole, participants are not aware that the double-click threshold can be changed in their computer (statement thirteen - I am aware that it is possible to change the speed at which a double click is recognized by the computer)

We can also draw the following more nuanced conclusions:

- While there is no clear-cut conclusion for statement six (I am comfortable with experimenting with computer settings in order to achieve the outcome I desire), the results lean heavily towards disagreement (47.37% versus 23.68%), and while the results do not meet the criteria for disagreement outlined at the start of this chapter, they nonetheless show that there are a considerable number of participants who do not feel comfortable when experimenting with their systems.
- While almost one in five of the participants in this study said they did not know that computers had a wide range of settings they could change, this is relatively low in comparison

to the number who did know. This suggests, with reference to previous literature that older users are becoming more familiar with what can actually be done with their systems. One in five represents a substantial proportion of the participant study however, and the need to support this group cannot be dismissed. (statement eight - I know that computers have a wide range of settings designed to make it easier for me to perform the tasks I need to perform)

- On the whole, almost half of the participants in the study felt comfortable with the idea of the control panel, but over thirty percent did not. The control panel is the access point to all of the computer's system configuration options. The contentious nature of the results suggests that, certainly for a substantial subset of users, that the current mechanisms for adjusting system functionality are suboptimal. (Statement nine – I understand what is meant by the term 'control panel')
- Over half of the participants did not know they could change the speed of the computer's cursor, but almost a thirty percent did. The results here lean strongly towards supporting the arguments for the framework, but are not without contention. (Statement twelve - I am aware that it is possible to change the speed at which my mouse cursor moves in response to my movements)

The following table summarises the expected versus actual results of these statements:

Number	**Statement Text**	**Expected**	**Actual**
5	I have a solid understanding of the range of options available for me with regards to configuring my computer	Disagree	Disagree
6	I am comfortable with experimenting with computer settings in order to achieve the outcome I desire	Disagree	Leaning to Disagree
8	I know that computers have a wide range of settings designed to make it easier for me to perform the tasks I need to perform	Disagree	Leaning to Agree
9	I understand what is meant by the term 'control panel'	Disagree	Contentious (Leaning to agree)
12	I am aware that it is possible to change the speed at which my mouse cursor moves in response to my movements	Disagree	Contentious (Leaning to disagree)
13	I am aware that it is possible to change the speed at which a double click is recognized by the computer	Disagree	Contentious (Leaning to disagree)

Table 26 – For Expected versus Actual (1 of 3)

Support for the framework with regards to this argument is thus not clear-cut. Even in those statements though where the results are either contentious or leaning away from the expected results, there are non-trivial numbers of participants on the other end of the scale. In conclusion, there is some support for this argument on the basis of the results gathered, but the argument cannot be considered to be accepted or rejected.

9.4.2 Older users do not know how to change settings

This argument is assessed in statements ten, eleven, fifteen, eighteen and nineteen. The following firm conclusions can be drawn from the results:

- On the whole, participants do not feel as if they can configure a computer if asked to do so. (Statement fifteen - If it was necessary for me to change the way my computer was set up, I would know how to do that)
- On the whole, participants do not understand the difference between application settings and system settings (Statement eighteen - I understand the difference between application-specific settings and system wide settings)
- On the whole, participants do not feel as if they can change the keyboard settings to improve their computing experience (Statement nineteen - I am aware of how to change the behaviour of my keyboard to make it more suitable for my own requirements)

The following more nuanced conclusions can be drawn:

- Half of the participants in the study do not know how to change the mouse speed settings, but over thirty percent do. This suggests that mouse configuration is considered to be something more useful on a day to day basis than changing the keyboard settings. However, while the statement is regarded to be contentious because of the relatively high agreement rate, it is still the case that fully half of the participants in the study reported that they do not know how to make this change. (Statement ten - I am aware how to change the behaviour of my computer's mouse to make it more suitable for my own requirements)
- It was expected that most users would know how to change the settings in their most familiar applications, but over half did not. This suggests that even in those cases where an application is configured sub-optimally, the majority of older users will not be able to make the changes needed to improve their experience. (Statement eleven - I know how to change the settings within my most familiar computer programs)

The following table summarises the expected versus actual results of these statements:

Number	Statement Text	Expected	Actual
10	I am aware how to change the behaviour of my computer's mouse to make it more suitable for my own requirements	Disagree	Contentious (Leaning to disagree)
11	I know how to change the settings within my most familiar computer programs	Agree	Contentious (Leaning to disagree)
15	If it was necessary for me to change the way my computer was set up, I would know how to do that	Disagree	Disagree
18	I understand the difference between application-specific settings and system wide settings.	Disagree	Disagree
19	I am aware of how to change the behaviour of my keyboard to make it more suitable for my own requirements	Disagree	Disagree

Table 27 – For Expected versus Actual (2 of 3)

Support then for this argument is strong – even in statement ten (I am aware how to change the behaviour of my computer's mouse to make it more suitable for my own requirements) where the results are contentious, fully half of the participants disagreed with the statement in line with expectations. Statement eleven (I know how to change the settings within my most familiar computer programs), while contrary to expectations, does not impact on the argument with regards to the tool as a mechanism for performing system corrections. Indeed, it suggests that the scope of the tool could be meaningfully expanded to include application level configuration also. Such expansions would encounter considerable technical challenges in the framework's current form, and are discussed in the chapter on future work.

9.4.3 Older users lack the confidence to make changes

This argument is assessed in statements one, three, seven, sixteen and seventeen. The following firm conclusions can be drawn from the results:

- On the whole, participants do not feel confident when determining the impact of changing their computer settings (Statement seven - I have confidence that I can correctly determine the impact of changing computer settings in most situations)
- On the whole, participants will work with an application as it is presented rather than attempt to optimise it (Statement sixteen - I usually work with an application as it is presented to me, without attempting to change the way it is set up)
- On the whole, participants experience anxiety when changing settings with which they are not familiar (Statement seventeen - I am anxious when changing a program setting with which I am not familiar).

The following more nuanced conclusion can be drawn:

- Roughly as many participants feel as if they understand the link between their actions and the outcomes as those who feel as if they do not. While the responses are contentious, there are a substantial number of users who feel as if they do not usually understand why their actions produce the computer output they do (Statement three - I usually understand the link between my actions and the outcome when working with a computer).

The following table summarises the expected versus actual results of these statements:

Number	Statement Text	Expected	Actual
1	I consider myself to be a novice computer user	Agree	Leaning towards disagree
3	I usually understand the link between my actions and the outcomes when working with a computer	Disagree	Contentious (Leaning towards agree)
7	I have confidence that I can correctly determine the impact of changing computer settings in most situations	Disagree	Disagree
16	I usually work with an application as it is presented to me, without attempting to change the way it is set up	Agree	Agree
17	I am anxious when changing a program setting with which I am not familiar.	Agree	Agree

Table 28 – For Expected versus Actual (3 of 3)

While most users either understood or responded as neutral to the statement regarding the link between their actions and outcomes, a 29% disagreement rate is considerable given how core building a mental model is for understanding complex systems. There were also fewer self-reported novices in this study group than was expected, but it seems likely that this is a consequence of recruitment through the SiDE user pool.

Aside from these caveats, support for this argument is strong showing that users are on the whole anxious when changing unfamiliar settings, lacking in confidence when assessing the impact of the changes they make, and being unwilling to optimise applications, preferring instead to work with the application as it is presented.

9.5 Conclusion

The results from this part of the study suggest strong support for the fundamental arguments at the core of the design of the ACCESS Framework. While it seems that older users are becoming more familiar with the possibilities inherent in the systems they use, it is still the case that self-reported measures of knowledge and confidence with regards to computer configuration are low. Two of the three fundamental arguments (on the whole, older users lack the knowledge and older users lack the confidence) of the tool thus stand up to examination, whereas the third is more contentious – it can be said that ‘some older users lack a knowledge of what can be changed’, but not that it is as substantial an issue as it has been in previous years, at least as far as this experimental group can be considered. However, terms such as ‘ease of access’ and ‘accessibility’ were shown during this study to have low levels of traction amongst their intended demographic, which suggests that even the term ‘accessibility’ could be more accessible.

While this is somewhat contrary to some of the available literature on the topic (c.f. Trewin, 2000; Hawthorn, 2003), most of that literature was published some years ago and the pace of adoption, and simple number of installations, of computer systems across the board have increased since then. It seems likely that the growing familiarity of older users with the capabilities of their systems is a simple consequence of increased exposure in both their own lives and the culture as a whole.

10 A Study of Reinforcement Learning in Access – Experimental Results

10.1 Introduction

The majority of the time spent by participants in the experimentation sessions was spent on computer based tasks, with those tasks being spread over two separate studies in a repeated measures design. After each task was completed in each study, participants were asked to rate the task in terms of ease, responsiveness of the computing equipment, and their ability to make changes to the computer to simplify the task in future. In addition, objective measures of performance were tracked for each of the tasks.

10.2 Participants

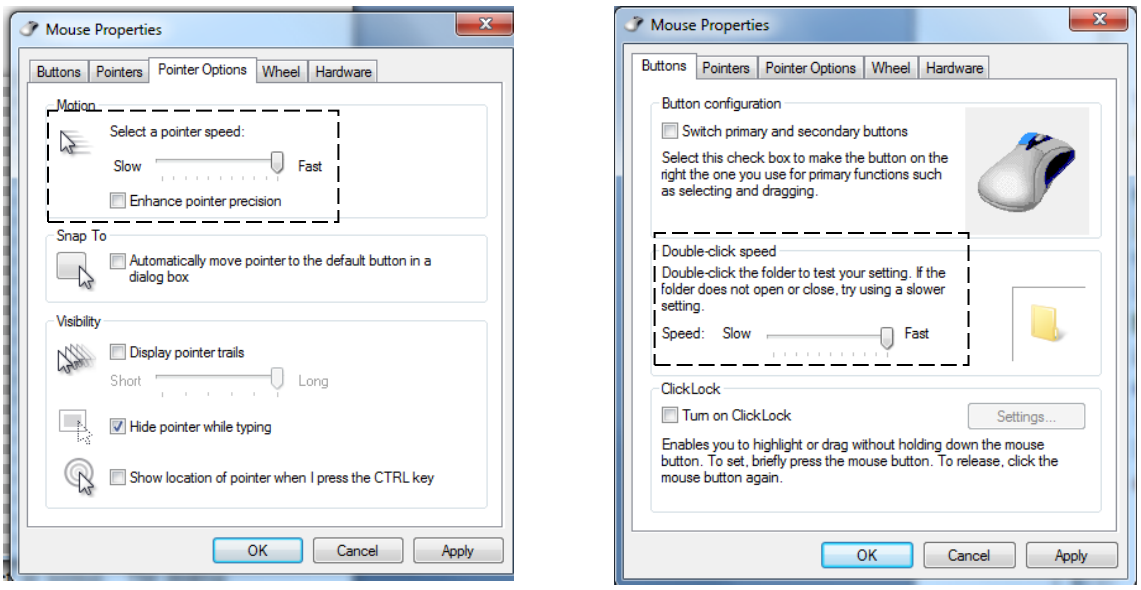
The same group of participants who completed the questionnaire in chapter nine were used for this study, comprising a group of individuals (N=38, M=15, F=23).

10.3 Materials

Participants were assessed in pairs, both participants using a Fujitsu Siemens Amilo Pro laptop running Windows XP and with wired mouse. Each of these laptops, in line with the experimental protocol discussed in a previous chapter, was configured to the extremes of what was permissible within the context of the operating system and computer system. In between each set of tasks, the computer was reset to this starting configuration. Specifically:

- The double click threshold was set to 200ms (fast)
- Resolution was set to 1024x768 (the maximum permitted by the hardware)
- Mouse speed was set to 20 (fast)
- Pointer precision was disabled

The control panel after these changes had been applied is demonstrated in figure 42:



Mouse Properties

Buttons | Pointers | Pointer Options | Wheel | Hardware

Motion
Select a pointer speed:
Slow — Fast
☐ Enhance pointer precision

Snap To
☐ Automatically move pointer to the default button in a dialog box

Visibility
☐ Display pointer trails
Short — Long
☑ Hide pointer while typing
☐ Show location of pointer when I press the CTRL key

OK | Cancel | Apply

Mouse Properties

Buttons | Pointers | Pointer Options | Wheel | Hardware

Button configuration
☐ Switch primary and secondary buttons
Select this check box to make the button on the right the one you use for primary functions such as selecting and dragging.

Double-click speed
Double-click the folder to test your setting. If the folder does not open or close, try using a slower setting.
Speed: Slow — Fast

ClickLock
☐ Turn on ClickLock
Settings...
Enables you to highlight or drag without holding down the mouse button. To set, briefly press the mouse button. To release, click the mouse button again.

OK | Cancel | Apply

Figure 42 - Mouse Control Panel for Study

In addition, mouse trails were switched off and pointer size was set as normal. The desktop background was set to one of the default Windows XP wallpapers, as shown in figure 43:

Figure 43 - Chosen Background on Testing System

10.4 Procedure

The research was conducted in two separate studies. The Double Click Study consisted of a single ACCESS plug-in and a pair of tasks focused primarily on double-clicking. The Multiple Plug-in Study consisted of five ACCESS plug-ins and five tasks spread across double clicking, mouse tracking, and targeting.

Each study used a repeated measures method, and the individuals in both studies were assigned randomly to one of two starting experimental conditions - the first condition involved attempting the tasks with the framework active, and then attempting them without the assistance of the framework.

For experimental condition two, participants were asked to perform the tasks first without assistance, and then again with the benefit of the framework.

10.4.1 The Double Click Study

In the Double Click Study, the framework was populated with a single plug-in, and participants were asked to complete a pair of tasks in which they attempted to double-click on static and moving targets. After successfully registering a double click, the testing application would move both the window and the mouse cursor to a random location on the screen. In both tasks, this process was repeated for five minutes or until the participant chose to abandon the task. These tasks are discussed in more detail in section 8.5 of this dissertation. The plug-in active for this study was the Double-click Detector as discussed in a chapter six. The first task involved clicking a static target, as in figure 44.

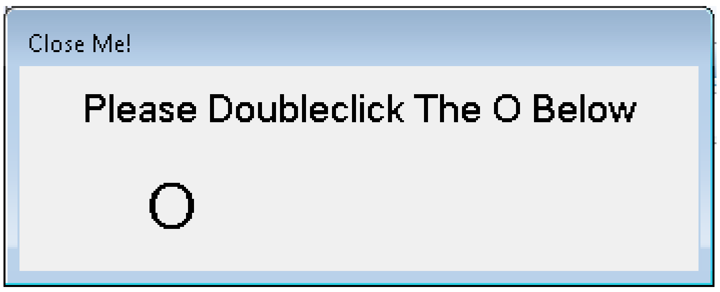


Figure 44 - Double Click Task 1

The second task involved double-clicking a moving target which bounced around the confines of a provided window as shown in figure 45.

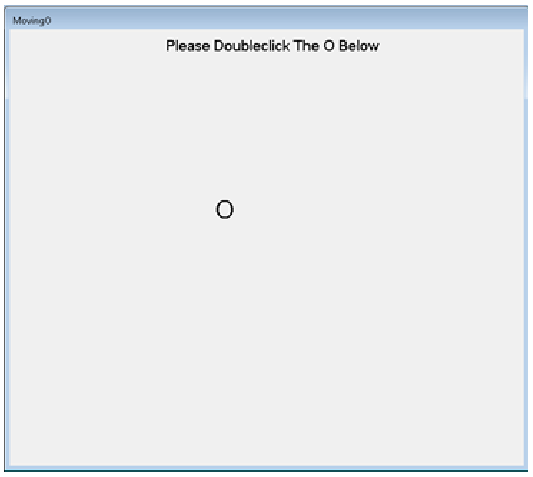


Figure 45 - Double Click Task 2

For the latter of these tasks it was anticipated that abandonment rates would be high, but that the additional complication would permit increased opportunities for the framework to pick up the need for corrective action.

Participants had the task explained to them before they attempted the task, and the way in which the ACCESS Framework would present its corrections to the users was explained and demonstrated by the experimenter. Participants did not receive a practice session, although this was as a result of time constraints rather than a conscious part of the experimental design. Participants were however encouraged to ask whatever questions they had of the researcher and clarification on tasks and questionnaire statements was available on request.

After each task, participants were asked to fill in a short, three item 5-point Likert scale questionnaire. The statements were about their subjective level of ease with regards to the task and their confidence in making adjustments to the system to make the task easier in future. For each task, the statements given for assessment were:

1. I feel that the task was easy to perform
2. I feel that the mouse was suitably responsive for the task
3. I know how to change the computer settings so that the mouse was more suited to perform this task

Objective figures regarding participant performance were also kept, showing the length of time between successful double clicks, and the number of successful clicks obtained during a testing period.

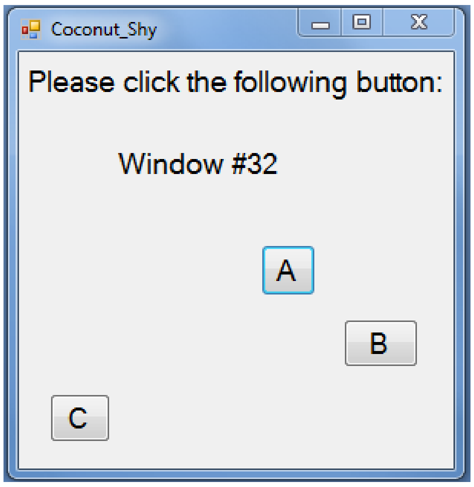
It was anticipated that exposure to the framework would increase both subjective and objective measures across both of these tasks.

10.4.2 Multiple Plug-in Study

In the Multiple Plug-in Study, the same experimental design was applied to a series of five tasks. The first two of these were as in Double Click Study, but the third, fourth and fifth tasks were unique to this study. The five tasks in order were:

- Double click a static target
    - As for the double-click study
- Double click a moving target
    - As for the double click study
- Click a static button as indicated on a separate sheet of paper
    - A screen shot of this task is provided in figure 38. The answer sheet given to candidates is provided in appendix six. Success was defined as the participant clicking the correct button for the number they were given.

  - Every time a button was clicked, the window and the pointer would be moved somewhere random on the screen, the position and size of buttons would change, and the number requested of candidates was changed.
- Click a moving button as indicated on a separate sheet of paper.
  - As with the task above, but this task also moved the buttons from side to side.
- Track a moving target, keeping the cursor as close to the centre as is possible.
  - Each iteration of this task involved participants tracking the circle until they had kept the pointer in the circle for at least five seconds, or until ten seconds had passed. At the end of either of these periods, the tasking software recorded how long they had managed to keep the pointer in the centre, and how long that iteration took to complete.

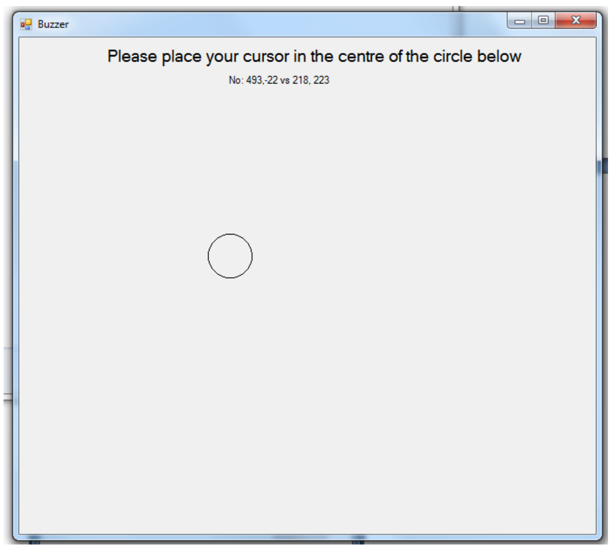


Please click the following button:

Window #32

Figure 46 - Screen Shot for Task 3/4 (Clicking Buttons)

Figure 47 shows the window for task 5.



Please place your cursor in the centre of the circle below

No: 493,-22 vs 218, 223

Figure 47 - Screen Shot for Task 5

For the third (click a static button) and fourth (click a moving button) tasks, the information needed was provided on a separate piece of paper to simulate a measure of real world distraction. Participants were required to direct their attention away from the screen and on to a separate sheet of paper before directing it once more to the screen. The intention of this was to increase the likelihood that participants would have mouse location problems without introducing artificial complications.

To support this wider range of tasks, five plug-ins were activated within the framework. The design of each these plug-ins is discussed in chapter six:

- Double click detector
- Double-back detector
- Missed click detector
- Pointer size plug-in
- Mouse trails plug-in

The plug-ins in this study were not tightly focused on a single task, and so it was anticipated that improvement across each of the tasks would be demonstrated, but the effect would be lower than was observed in the Double Click Study.

The experimental protocol of the Multiple Plug-in Study was identical to that of the Double Click Study, consisting of a repeated measures design of five minute tasks, the completion of each being followed by a three item questionnaire.

10.5 Results for the Double Click Study

10.5.1 Objective Results for the Double Click Study

Times and means are counted only for successfully registered double-clicks. Unsuccessful attempts to double-click are not factored into results, and were not recorded during the study. Section 10.7 later in this chapter discusses how reinforcement was applied by the framework. Data corruption invalided three of the participant measures, but the results for the remaining observations (N=35) are shown in table 29:

Tasks	Number of Successful Clicks (Mean)	Time Between Successful Clicks (Mean)	Abandonment Rate
1 (With framework)	59.4	9.239s	5%
1 (Without framework)	34.7	26.385s	100%
2 (With framework)	1.51	11.828s	0%
2 (Without framework)	0.69	26.756s	100%

Table 29 - Double Click Study (with and without)

Figure 48 shows the number of clicks registered for each participant in the two experimental conditions for the static target task:

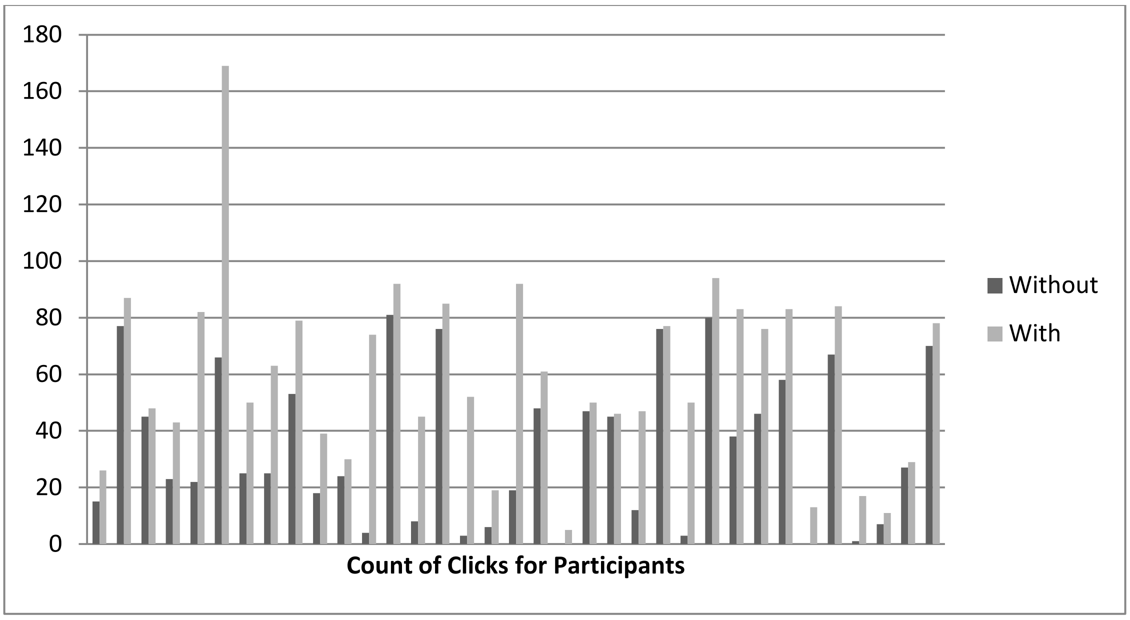


Figure 48 - Count of Clicks for Double Click Study Static Target Task

Figure 49 shows the number of clicks registered for each participant in the two experimental conditions for the moving target task:

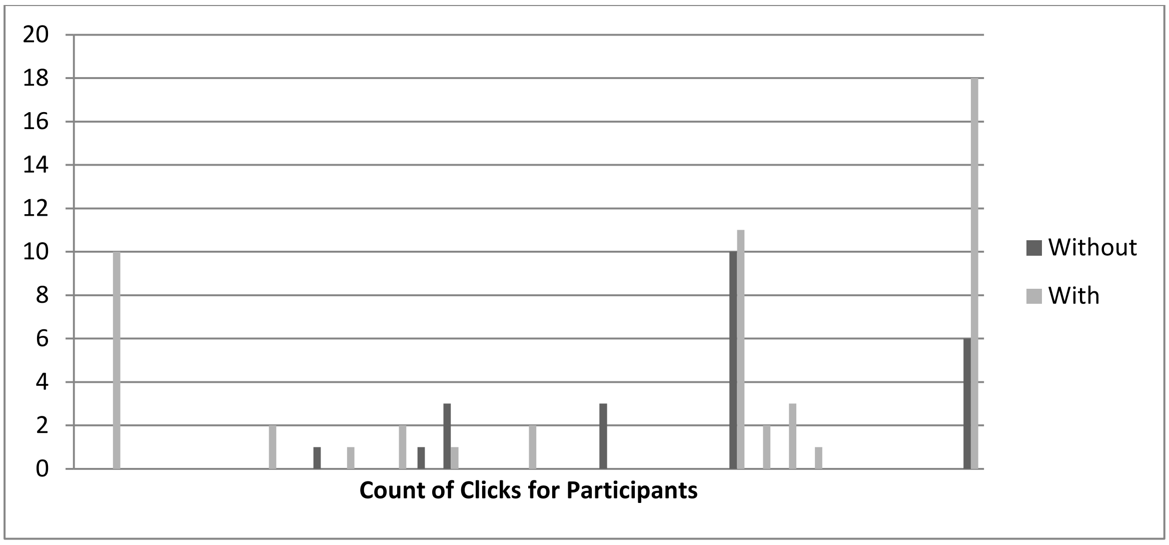


Figure 49 - Count of Clicks for Double Click Study Moving Target Task

The low number of successful clicks registered in task two in both conditions shows that there is no meaningful improvement from the framework in terms of the average or speed. However, for the static clicking task there is a significant difference between both the number of double clicks recorded with the framework active, and the time taken to perform those double clicks. Within the context of easing the double-click threshold only within this experimental trial, the objective measures of performance show considerable support for the value of the framework. For the first task, the difference in performance with regards to number of clicks is considered statistically significant (P<0.01, DF=34, T=6.13), and likewise for speed (p=0.0375, DF=34, T=1.83), using a one tailed student t-test. For the number of clicks in the moving target task, the results are also statistically significant (P<0.01, DF=34, T=3.18), and likewise for speed (P<0.02, DF=34, T=2.95). For task two, the benefit of the framework is that it permitted more individuals to successfully manage a double-click. Without correction only six individuals managed any clicks at all, whereas in the second condition eleven managed. Thus while the framework did not make the task especially easier for any of the participants, it did allow more participants to achieve an otherwise impossible goal.

10.5.1.1 Subjective Results for Double Click Study

For the statement 'I feel that the task was easy to perform', the following results were gathered across both tasks:

Task	**Agreement Rate**	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual Result**	**Standard Deviation**
Double-Clicking a stationary button (With)	60.53%	13.16%	26.32%	SA	Agree	Agree	1.11
Double-Clicking a stationary button (Without)	34.21%	44.74%	21.05%	SD	Disagree	Contentious	1.30
Double-Clicking a moving button (With)	7.89%	92.11%	2.63%	SD	Contentious	Disagree	1.07
Double-Clicking a moving button (Without)	2.63%	97.37%	0%	SD	Disagree	Disagree	0.75

Table 30 - Double Click Study - I feel that the task was easy to perform

For the statement 'I feel that the mouse was suitably responsive for the task', the following results were gathered:

Task	Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual Result	Standard Deviation
Double-Clicking a stationary button (With)	47.37%	23.68%	28.95%	N	Agree	Leaning towards agree	1.17
Double-Clicking a stationary button (Without)	23.68%	60.53%	15.79%	N	Disagree	Disagree	1.18
Double-Clicking a moving button (With)	10.53%	76.32%	13.16%	N	Contentious	Disagree	1.22
Double-Clicking a moving button (Without)	7.89%	89.47%	2.63%	N	Disagree	Disagree	0.97

Table 31 - Double Click Study - I feel that the mouse was suitably responsive

For the statement 'I know how to change the computer settings so that the mouse was more suited to perform this task', the following results were gathered:

Task	**Agreement Rate**	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual Result**	**Standard Deviation**
Double-Clicking a stationary button (With)	21.05%	65.79%	13.16%	SD	Disagree	Disagree	1.46
Double-Clicking a stationary button (Without)	23.68%	68.42%	7.89%	SD	Disagree	Disagree	1.74
Double-Clicking a moving button (With)	15.79%	78.95%	5.26%	SD	Disagree	Disagree	1.52
Double-Clicking a moving button (Without)	10.53%	84.21%	5.26%	SD	Disagree	Disagree	1.33

Table 32 - Double Click Study - I know how to change the computer settings

Comparisons of the averages for experimental conditions for task one across of each of these statements yields the following table:

Statement	**Average (With)**	**Average (Without)**	**Statistical Significance**	**Degrees of Freedom**	**T Stat**
I feel that the task was easy to perform	3.68	2.66	P < 0.001	37	5.65
I feel that the mouse was suitably responsive.	3.37	2.53	P < 0.01	37	3.16
I know how to change the computer settings.	2.03	2.03	P < 0.36	37	0.35

Table 33 - Statistical Significance of Subjective Feedback for Task One

For task two, the results table is as follows:

Statement	Average (With)	Average (Without)	Statistical Significance	Degrees of Freedom	T Stat
I feel that the task was easy to perform	1.39	1.16	P < 0.05	37	1.84
I feel that the mouse was suitably responsive.	1.63	1.47	P < 0.1	37	1.35
I know how to change the computer settings.	1.76	1.58	P < 0.07	37	1.48

Table 34 - Statistical Significance of Subjective Feedback for Task Two

Graphical comparisons of the distribution of responses on paired trials show that for both statements one and two there is an improvement in subjective measures of performance and responsiveness

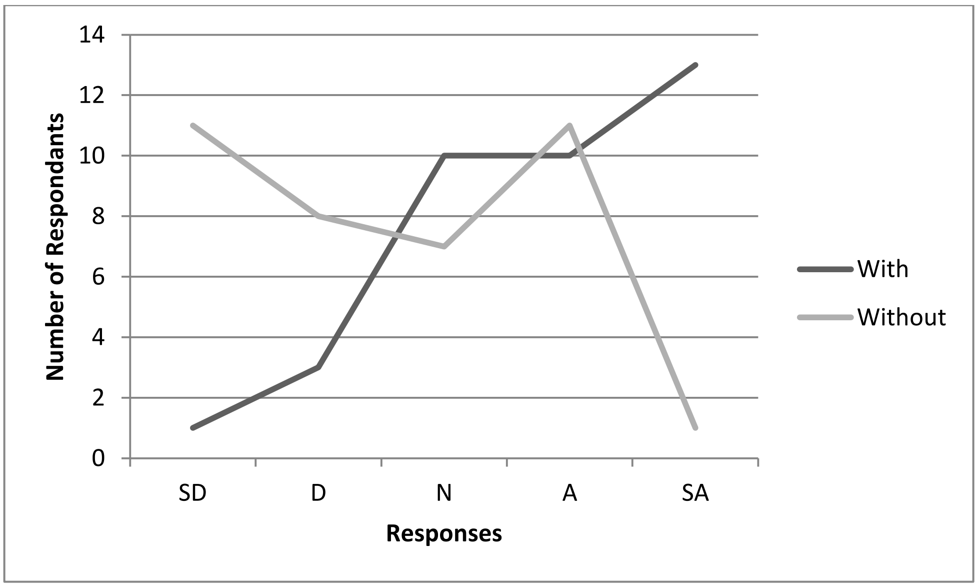


Figure 50 - The Task was Easy to Perform (Task One)

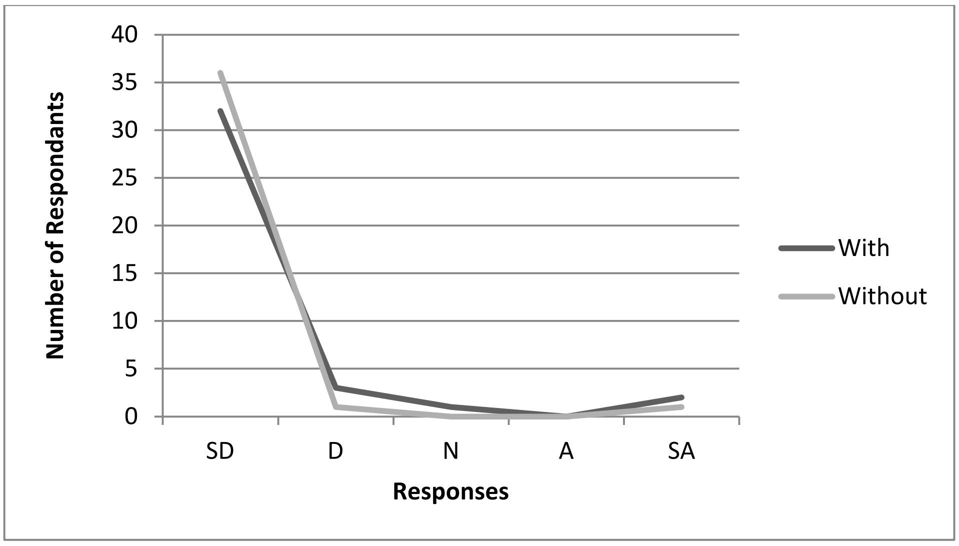


Figure 51 – The Task was Easy to Perform (Task Two)

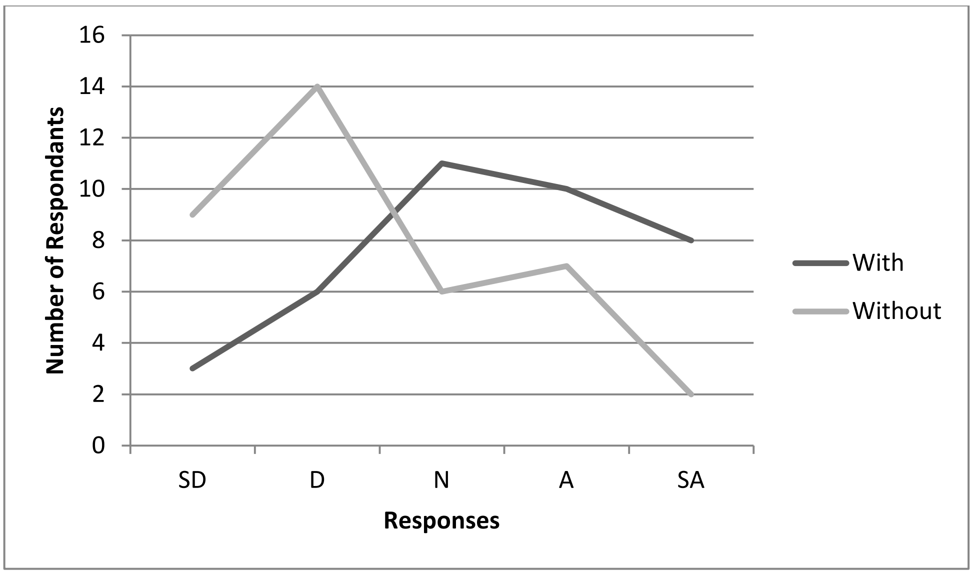


Figure 52 - The Mouse Was Suitability Responsive (Task One)

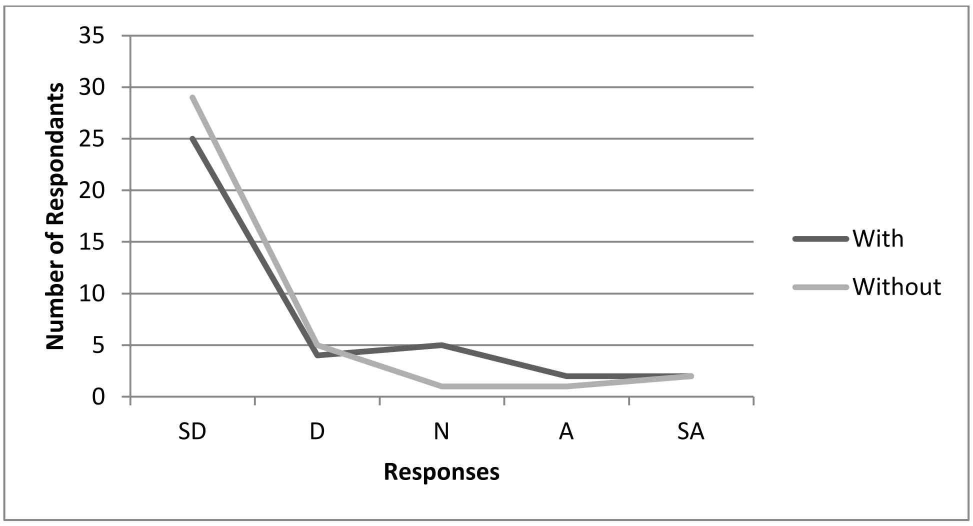


Figure 53 - The Mouse Was Suitability Responsive (Task Two)

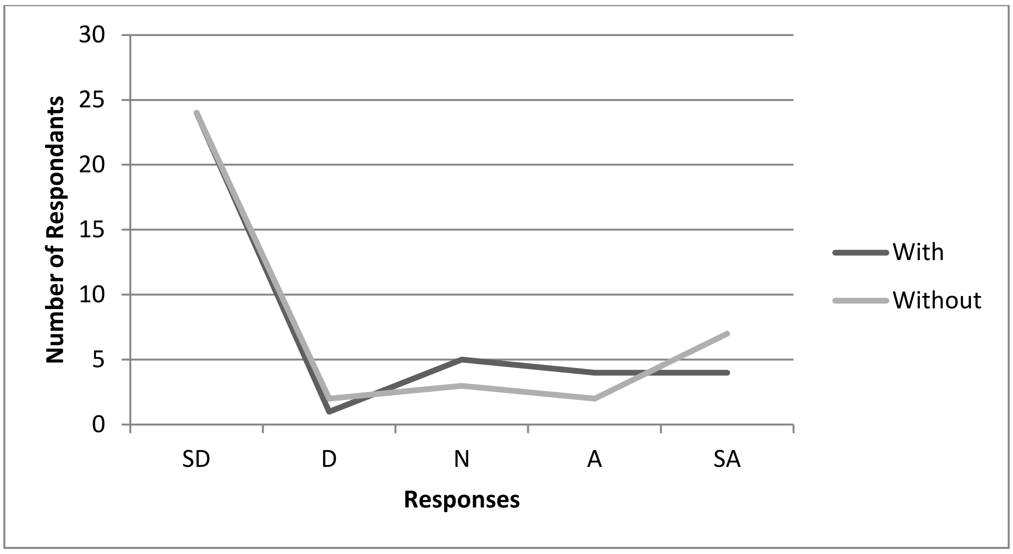


Figure 54 - I know how to change the computer settings (Task One)

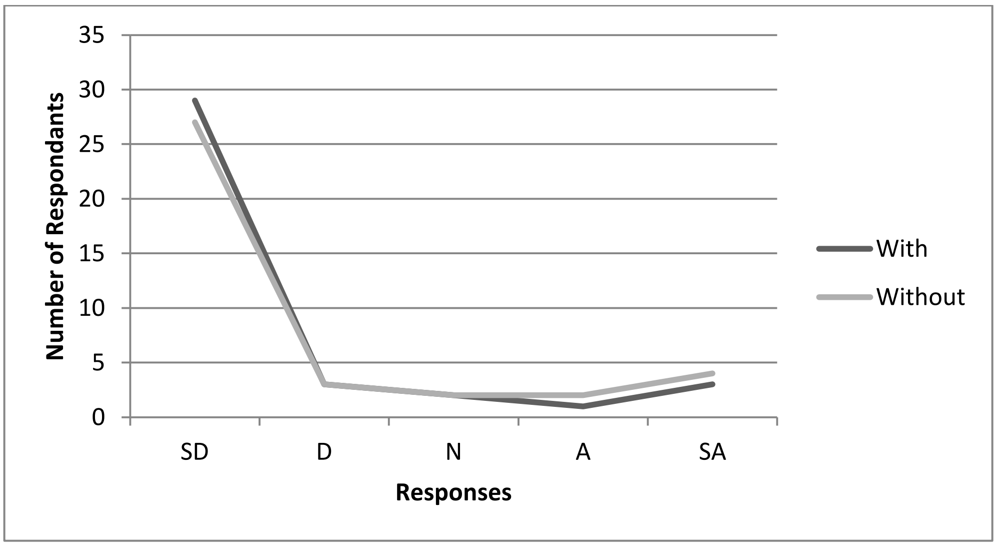


Figure 55 - I know how to change the computer settings (Task Two)

10.5.2 Discussion of results

The results permit for the following conclusion to be drawn:

- Adaptive correction of the double click threshold has a positive impact on the recorded subjective and objective measures for both tasks.

Reported values of user confidence with regards to computer configuration demonstrate some noise, perhaps due to individuals reporting higher confidence as a result of the framework offering a mechanism for configuration. Overall classification of the results though remains unchanged from condition to condition, showing that on the whole participants in the trial do not consider themselves able to make the changes to the system that would improve their performance in the tasks.

Results from the first task show promising results, with an average improvement of 306.87%, with a high degree of statistical significance across participants. In two cases, participants recorded an improvement rate in excess of 1500%, showing that a task that was previously extremely difficult was made tractable by successive adaptations by the framework.

However, abandonment rates for the second task (double clicking a moving target) were 100% in both experimental conditions - all participants requested abandonment due to high levels of frustration with that particular exercise. Results from the second task show that while the task did not become much easier for any of the participants, the application of the framework allowed for more participants to register a successful double click. This effect is considered to be statistically significant by the analysis run on the figures. In addition, there is a small improvement in the subjective measure of ease ($P < 0.05$) and responsiveness. While the latter is not significant at the $P < 0.05$, it is significant at $P < 0.1$.

When the right plug-in is matched to the right task, the improvement rates associated with the framework can be substantial. Most operating systems will not come configured at the extremes of the operating system as the test machine's. It is clear though that in sub-optimal configurations the framework can pick up the need for corrective measures and make those corrections. In extreme cases, this can turn an extremely difficult task into something more achievable. Consistent with the literature and the study results discussed in the previous chapter, there is strong support for the need for the framework at a user level – disagreement rates for the statement 'I know how to change the computer settings so the mouse is more responsive for the task' are greater than sixty five percent across each task.

10.6 Multiple Plug-in Study

10.6.1 Results for Multiple Plug-in Study

10.6.1.1 Objective Results for Multiple Plug-in Study

The following table summarises the difference in objective performances between the two experimental conditions:

Tasks	**Mean (Completion)**	**Speed**
Stationary Button Double Click (with framework)	61	6.935s
Stationary Button Double Click (without framework)	46	15.617s
Moving Button Double Click (with framework)	1.5	10.1.795s
Moving Button Double Click (without framework)	0.85	12.326s
Look away and click stationary button (With framework)	48	6.622s
Look away and click stationary button (Without framework)	49	6.548s
Look away and click moving button (With framework)	32	10.937s
Look away and click moving button (Without framework)	27	12.403s
		Accuracy
Track moving circle (with framework)	15	68.5%
Track moving circle (without framework)	13	66.4%

Table 35 - Multiple Plug-in Study (with and without)

For task one, the results have statistical significance for both the number of clicks (P<0.01, DF=34, T=5.12) and the speed of double-clicking (P<0.01, DF=34, T=3.76), consistent with the results obtained in the Double Click Study. For task two, the results are once again statistically significant for number of clicks (P<0.01, DF=34, T=2.475), but not for speed. Once again, the number of participants registering successful clicks (4 without correction, 8 with) is the distinguishing feature between the two experimental conditions.

For task three, there is no statistical significance in the results. The framework did not assist in the performance of this task. For task four, there is statistical significance for the number of clicks registered (P<0.03, DF=34, T=2.02), but not for speed. Similarly, for task five there is no statistical significance in the analysis of the figures for either number of successful completions or accuracy.

10.6.1.2 Subjective Results for Multiple Plug-in Study

For the statement 'I feel that the task was easy to perform', the following results were gathered across both tasks:

Task	**Agreement Rate**	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual Result**	**Standard Deviation**
Static Button Double Click (With)	73.68%	5.26%	21.05%	A	Agree	Agree	1.07
Static Button Double Click (Without)	47.37%	26.32%	26.32%	N	Disagree	Contentious	1.25
Moving Button Double Click (With)	5.26%	86.84%	7.89%	SD	Disagree	Disagree	0.76
Moving Button Double Click (without)	5.26%	89.47%	5.26%	SD	Disagree	Disagree	1.11
Look away and click stationary button (With)	89.47%	2.63%	7.89%	SA	Agree	Agree	0.67
Look away and click stationary button (Without)	81.58%	5.26%	13.16%	SA	Contentious	Agree	0.77
Look away and click moving button (With)	60.53%	15.79%	23.68%	A	Agree	Agree	1.06
Look away and click moving button (Without)	36.84%	28.95%	34.21%	N	Contentious	Contentious	1.00
Track moving circle (with)	2.63%	81.58%	15.79%	SD	Disagree	Disagree	1.01
Track moving circle (without)	10.53%	84.21%	5.26%	SD	Disagree	Disagree	1.18

Table 36 - Multiple Plug-in Study - I feel that the task was easy to perform

For the statement 'I feel that the mouse was suitably responsive for the task', the following results were gathered:

Task	**Agreement Rate**	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual Result**	**Standard Deviation**
Static Button Double Click (With)	68.42%	13.16%	18.42%	A	Agree	Agree	1.07
Static Button Double Click (Without)	47.37%	26.32%	26.32%	N	Disagree	Contentious	1.25
Moving Button Double Click (With)	13.16%	65.79%	21.05%	SD	Contentious	Disagree	1.04
Moving Button Double Click (without)	7.89%	84.21%	7.89%	SD	Disagree	Disagree	1.22
Look away and click stationary button (With)	92.11%	2.63%	5.26%	SA	Agree	Agree	0.77
Look away and click stationary button (Without)	73.68%	7.89%	18.42%	SA	Contentious	Agree	0.92
Look away and click moving button (With)	52.63%	15.79%	31.58%	N	Agree	Agree	1.03
Look away and click moving button (Without)	28.95%	28.95%	42.11%	N	Disagree	Contentious	0.95
Track moving circle (with)	23.68%	52.63%	23.68%	SD	Agree	Disagree	1.28
Track moving circle (without)	13.16%	71.05%	15.79%	SD	Disagree	Disagree	1.26

Table 37 - Multiple Plug-in Study - I feel the mouse was suitably responsive

For the statement 'I know how to change the computer settings so that the mouse was more suited to perform this task', the following results were gathered:

Task	**Agreement Rate**	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual Result**	**Standard Deviation**
Static Button Double Click (With)	31.58%	60.53%	7.89%	SD	Disagree	Contentious	1.55
Static Button Double Click (Without)	26.32%	65.79%	7.89%	SD	Disagree	Contentious	1.57
Moving Button Double Click (With)	18.42%	76.32%	5.26%	SD	Disagree	Disagree	1.34
Moving Button Double Click (without)	13.16%	81.58%	5.26%	SD	Disagree	Disagree	1.35
Look away and click stationary button (With)	34.21%	60.53%	5.26%	SD	Disagree	Contentious	1.59
Look away and click stationary button (Without)	28.85%	65.79%	5.26%	SD	Disagree	Contentious	1.55
Look away and click moving button (With)	26.32%	63.16%	10.53%	SD	Disagree	Contentious	1.50
Look away and click moving button (Without)	15.79%	73.68%	10.53%	SD	Disagree	Disagree	1.36
Track moving circle (with)	15.79%	78.95%	5.26%	SD	Disagree	Disagree	1.43
Track moving circle (without)	15.79%	81.58%	2.63%	SD	Disagree	Disagree	1.25

Table 38 - Multiple Plug-in Study - I know how to change the computer settings

Comparisons of the averages for task one across of each of these statements yields the following table:

Statement	**Average (With)**	**Average (Without)**	**Statistical Significance**	**Degrees of Freedom**	**T Stat**
I feel that the task was easy to perform	3.97	3.32	P < 0.00	37	3.69
I feel that the mouse was suitably responsive.	3.79	3.18	P < 0.00	37	3.46
I know how to change the computer settings.	2.42	2.16	P = 0.16	37	1.02

Table 39 - Statistical Significance of Subjective Feedback for Multiple Plug-in Study, Task One

For task two, the results table is as follows:

Statement	**Average (With)**	**Average (Without)**	**Statistical Significance**	**Degrees of Freedom**	**T Stat**
I feel that the task was easy to perform	1.47	1.26	P = 0.36	37	0.36
I feel that the mouse was suitably responsive.	1.95	1.39	P < 0.05	37	1.77
I know how to change the computer settings.	1.79	1.63	P < 0.05	37	1.78

Table 40 - Statistical Significance of Subjective Feedback for Multiple Plug-in Study, Task Two

For task three, the results table is as follows:

Statement	**Average (With)**	**Average (Without)**	**Statistical Significance**	**Degrees of Freedom**	**T Stat**
I feel that the task was easy to perform	4.61	4.32	P < 0.05	37	1.82
I feel that the mouse was suitably responsive.	4.45	4.08	P < 0.01	37	2.67
I know how to change the computer settings.	2.34	2.24	P = 0.29	37	0.56

Table 41 - Statistical Significance of Subjective Feedback for Multiple Plug-in Study, Task Three

For task four, the results table is as follows:

Statement	Average (With)	Average (Without)	Statistical Significance	Degrees of Freedom	T Stat
I feel that the task was easy to perform	3.50	3.03	P < 0.00	37	3.48
I feel that the mouse was suitably responsive.	3.59	2.95	P < 0.01	37	2.60
I know how to change the computer settings.	2.24	1.87	P < 0.02	37	2.11

Table 42 - Statistical Significance of Subjective Feedback for Multiple Plug-in Study, Task Four

For task five, the results table is as follows:

Statement	Average (With)	Average (Without)	Statistical Significance	Degrees of Freedom	T Stat
I feel that the task was easy to perform	1.75	1.53	P = 0.4	37	0.26
I feel that the mouse was suitably responsive.	2.45	1.84	P < 0.02	37	2.19
I know how to change the computer settings.	1.76	1.79	P = 0.5	37	0.00

Table 43 - Statistical Significance of Subjective Feedback for Multiple Plug-in Study, Task Five

Thus, statistical significance is shown almost across the board for subjective ratings - the only measures where statistical significance at $P < 0.05$ are not observed are in the ease of the double-clicking a moving target task (contrary to the results of the Double Click Study), and in the sense that individuals could make changes to the system in tasks one and three. A full graphical breakdown of the subjective feedback may be found in the appendices.

10.7 Reinforcement

The final quantitative aspect of the framework that was under investigation was the nature of reinforcement itself. While it was seen as an understandable mechanic, the framework tracked the number of times plug-ins were reinforced to see whether or not users were making use of the options they were provided, as well as the number of times on average each plug-in identified a need for a correction.

Table 42 shows the details for the double-click study, and the double-click plug-in that was the focus of that research.

Plug-In	**Times Invoked**	**Positively Reinforced**	**Positively Punished**	**Auto Reinforced**
	Count	Count	Count	Count
Double-Click	270	135	39	96
	Mean	**Mean**	**Mean**	**Mean**
Double-Click	6.42	3.52	1.05	2.52
	Percentage	**Percentage**	**Percentage**	**Percentage**
Double-Click	100%	44.68%	15.98%	39.34%

Table 44 - Plug-In Invocations for Double-Click Study

Table 43 shows the details for the Multiple Plug-Ins study, and the details for the five plug-ins that formed the suite for that research:

Plug-In	**Times Invoked**	**Positively Reinforced**	**Positively Punished**	**Auto Reinforced**
	Count	**Count**	**Count**	**Count**
Double-Click	343	190	55	98
Mouse-Trails	108	10	29	69
Double Back	44	8	4	32
Pointer Size	81	14	12	55
Missed Clicks	51	10	6	35
	Mean	**Mean**	**Mean**	**Mean**
Double-Click	9.02	5.00	1.44	2.57
Mouse Trails	2.84	0.26	0.76	1.81
Double Back	1.15	0.21	0.10	0.84
Pointer Size	2.13	0.36	0.31	1.44
Missed Clicks	1.34	0.26	0.15	0.92
	Percentage	**Percentage**	**Percentage**	**Percentage**
Double-Click	54.70%	55.39%	16.03%	28.57%
Mouse Trails	17.22%	9.26%	26.85%	63.89%
Double Back	7.02%	18.18%	9.09%	72.73%
Pointer Size	12.92%	17.28%	14.81%	67.90%
Missed Clicks	8.13%	19.61%	11.76%	68.63%

Table 45 - Plug-In Invocations for Multiple Plug-ins Study

Table 44 shows the collapsed total for types of reinforcement for the multiple plug-ins study:

Positive Reinforcement	Positive Punishment	Auto Reinforcement
23.95%	15.71%	60.34%

Table 46 - Average Reinforcements for Multiple Plug-in Study

Plug-ins were permitted to auto accept after a correction interval had passed without the user registering a reinforcement. Positive reinforcement and auto accepted reinforcement have the same impact in terms of the effect on the operating system (in both cases, the change is committed as permanent), but only positive reinforcement alters the weighting and internal thresholds of a plug-in. Thus, in a total of 84.29% of corrections made, the correction was committed to the system. In less than 16% of corrections was it positively punished, and anecdotal evidence gathered during the trials suggest that some of those reinforcements were as a result of the participant not seeing the direct link between their current action and the recommended corrections.

However, a limitation of this study is that it is not possible to tell from the gathered data whether corrections were ‘intentionally’ left to auto accept or whether they were as a result of the participant simply not noticing the framework’s request for reinforcement. The figures gathered regarding intrusiveness of the framework show that participants on the whole did not feel that the framework was intrusive, and this may be a consequence of the fact that their attention was focused too tightly on the tasks themselves to notice the correction window when it was presented. In 18.4% of the sessions, no reinforcement at all was observed in the experimental data – all other sessions show at least one instance of reinforcement being performed. Discussion of the possibility that this was due to attention being focused on the performing of the requested task is discussed in chapter twelve.

From the figures alone, it is clear that individuals positively reinforce more often than they positively punish, and that they permit the system to auto accept more often than both other options combined. Additionally, it shows that individuals will not only reinforce the framework when the effect of a correction is obvious (such as the pointer size changing), they will also reinforce it when the impact of the correction is largely invisible (such as with changing the double-click threshold).

A hypothesis of this study was that individuals would each reinforce plug-ins differently, and that each would end up with a different profile of corrections as a result. Table 47 shows the standard deviation associated with the number of invocations of plug-ins.

	Missed Clicks	Double Clicks	Mouse Trails	Pointer Size	Double Back
Mean	1.34	9.02	2.84	2.13	1.15
Std. Dev	1.08	3.48	2.47	4.77	1.06

Table 47 - Standard Deviation of Invocations for Multiple Plug-ins study

Table 48 in turn shows the standard deviation between numbers and types of reinforcement:

	Positively Reinforced	**Positively Punished**	**Auto Reinforced**
Mean	6.15	1.52	9
Std. Dev	6.84	2.39	9.74

Table 48 - Standard Deviation of Conditioning for Multiple Plug-ins Study

Taken together, this shows a high degree of variability in the plug-ins that users were provided, and a similarly high degree of variability in how they applied the conditioning. The consequences of this is that each individual will be extremely likely to end up with a portfolio of accessibility corrections that are unique to them.

It seems likely that this auto-accepting of corrections is simultaneously an expression of the status quo effect (Kahneman et al, 1991) and silent consent for the adjustments the framework was making – in cases where the altered settings were unwelcome or unpleasant, that is likely to be sufficiently jarring to break someone out of their focus on a task. This is supported to an extent by the literature showing older adults find it difficult to ignore distracting information (Connelly and Hasher, 1993; Kotary and Hoyer, 1995). This is a hypothesis that is discussed in chapter twelve.

10.7.1 Discussion of Results

The results permit for the following conclusions to be drawn:

There is a statistically significant effect with regards to the following measures:

- Number of successful double clicks registered in the static double clicking task
- Time between successful double clicks in the static double clicking task
- Number of successful double clicks registered in the moving double clicking task
- Number of successful clicks in the moving button with attention distraction task.

Improvement that is statistical significant to 0.05 is observed in the following subjective measures:

- Measures of both ease and mouse responsiveness in double clicking a static target
- Measures of mouse responsiveness only for the moving double click task
- Measures of both ease and mouse responsiveness in task three
- Measures of both ease and mouse responsiveness in task four
- Measures of both ease and mouse responsiveness for the mouse tracking task.

It is possible that for the mismatch between subjective and objective improvement of the tasks (such as in task three and task five which have no statistically significant objective impact) that the improvement can be explained by a placebo effect. It is also possible that the additional cognitive overhead required to read the feedback from the framework, consider its implications and try out its corrections incurs an additional time debt that obscures any statistically significant improvement in the results. Such a conclusion can only be speculative with the data available from this study since relevant data regarding time spent administering the framework was not gathered during the experiment.

Additionally, it is not possible in this testing scenario to explicitly link plug-ins to impact – a wider range of corrective measures were performed by the framework during this study than in the Double Click Study, and there is no guarantee that corrective behaviour would link to current tasks. Participant comments occasionally questioned the corrective decisions taken by the framework because they did not match up to their immediate actions.

Participant feedback also suggested that certain corrective actions, while algorithmically decided by the framework on the basis of previous interactions, were occasionally unhelpful for current tasks. The most common example of this was when mouse-trails were enabled during task five. Participants felt that this correction, while helpful in locating the mouse, made it more difficult to perform the tracking exercises.

Individual plug-ins must carry the burden of demonstrating worth, and the double-click detector plug-in demonstrates that improvement can be substantial when a good match is found between intention and functionality. The objective and subjective measures in this study were aimed at analyzing the impact of the selected plug-ins, each of which was written and designed to assess a particular interaction need. From the results, it is clear to see that a number of statistically significant effects have emerged from the application of the tool.

In those tasks where no objective improvement is recorded despite improved subjective measures, it is hypothesized that the provision of this suite of tools simply makes for a more pleasant experience, if not necessarily a more efficient setup. This is a conclusion that is supported by the evidence as well as subjective feedback from participant debriefing. Several participants made special comment on the behaviour of the mouse cursor when they had been successful in performing one of the tasks, complaining that it was difficult to find the mouse cursor against the sub-optimal desktop wallpaper. These participants however found that increased pointer size and mouse trails assisted in this problem – while this may not have resulted in an increase in performance, increased user satisfaction would explain the results gathered as well as the feedback recorded in the after-task questionnaire discussed in the next chapter.

10.8 Conclusion

Demonstrating objective improvements in user performance was secondary in this study to the task of analyzing the ability of the framework to assess the need for corrections and then making those corrections whilst also providing users with the tools to meaningfully interact with the framework. It is clear from the results gathered that users feel that, subjectively, the tasks are easier when the framework is active than when it is not even in those cases when there is no corresponding increase in objective performance. It is hypothesized that the additional administrative tasks associated with the framework incur a time debt that is not reflected in the figures, but evidence for this hypothesis is derived only from anecdotal discussions with participants in the study, as well as supporting evidence in the literature regarding cognitive burdening (Newell et al, 2003; Hawthorn, 2000).

Two of the tasks demonstrated no statistically significant difference in objective measures. In both these tasks, there is a statistically significant effect with regards to subjective measures from participants. This difference is perhaps explained by accepting the provision of this suite of plug-ins results in a generally more pleasant user experience as a result of minor interaction impairments being smoothed out by the framework. In short, it is the hypothesis of this thesis that the results suggest that even if the tasks don't become objectively easier to perform, corrections by the framework result in less frustration on the part of users.

However, in three of the five tasks, objective improvement is recorded, and this argues strongly for the validity of the framework in making meaningful and useful corrections on behalf of users.

11 A Study of Reinforcement Learning in Access – Framework Results

11.1 Introduction

The final set of results gathered as part of this pair of studies was subjective feeling regarding the framework itself. After the completion of all computer tasks, participants were asked to complete a final fifteen item questionnaire. This questionnaire addressed the suitability of the reinforcement training metaphor and the perception of participants with regards to the framework's value as a configuration tool. As with the previous subjective data, results were expressed on a five point Likert scale, and for the purposes of analysis were categorised as agreement, disagreement or neutrality. Overall assessment of the data is done on the same basis as is outlined in previous chapters.

11.2 After Study Questions

11.2.1 Statement One: I understand the concept of 'liking' and 'disliking' changes made to the system as I performed the tasks

It was not anticipated that the concept of expressing a like or dislike would be foreign to participants, but this metaphor as applied to the user interface is not one with which most participants will be familiar. Settings in a computer in some cases may permit the user some time to confirm a change before it is committed (for example, changing the screen resolution), but this metaphor has not before been consistently applied to user interface configuration. It was anticipated that the metaphor is suitably tractable, and that agreement with this statement would be high, which was confirmed by the experimental data, as is demonstrated in the following table:

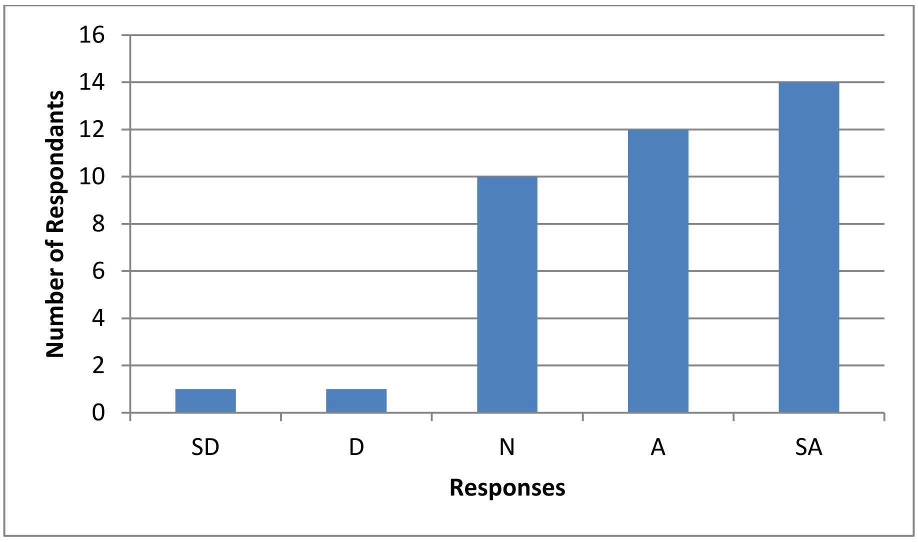


Figure 56 - Distribution of Results for Statement 1

The reinforcement system is a core mechanic of the framework, and tractability is virtually mandated in order for it to be effective from the perspective of users. The high levels of agreement and low levels of disagreement recorded provide support for it as a suitable primary mechanic for system configuration.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
68.42%	5.26%	26.32%	SA	Agree	Agree	1.05

Table 49 - Post-study Statements 1

11.2.2 Statement Two: I felt that the need for me to indicate my likes and dislikes was intrusive

There is a large amount of information that computer systems present for user attention, and one of the potential drawbacks of the ACCESS Framework is that it is an additional, intrusive call on user attention. Were that true, the intrusiveness of user prompts would be an unwelcome deterrent to adoption of the tool. The framework presents information as part of a non-modal dialog, and as such it can remain conditioned or not as the user desires without the participant having to formally acknowledge each time. As such it was hypothesized that this would not be seen as an interruption with regards to participants performing the tasks.

Within the context of an experimental study, the time between reinforcement periods must be shorter than would be possible in real world scenarios. While high agreement with this statement would not necessarily invalidate the worth of the inform mechanic, it would stress the importance of appropriate testing of the timing periods before deployment.

However, disagreement with the statement was high even considering the artificially short period between corrections. In a proper deployment, a longer period of time for assessment of corrective periods could be an appropriate change. This should be done though in line with existing literature regarding the impact of interruptions on flow of concentration and frustration amongst users. The results gathered from this study strongly suggest however that the fundamental design, from the perspective of intrusiveness, does not require substantial re-engineering.

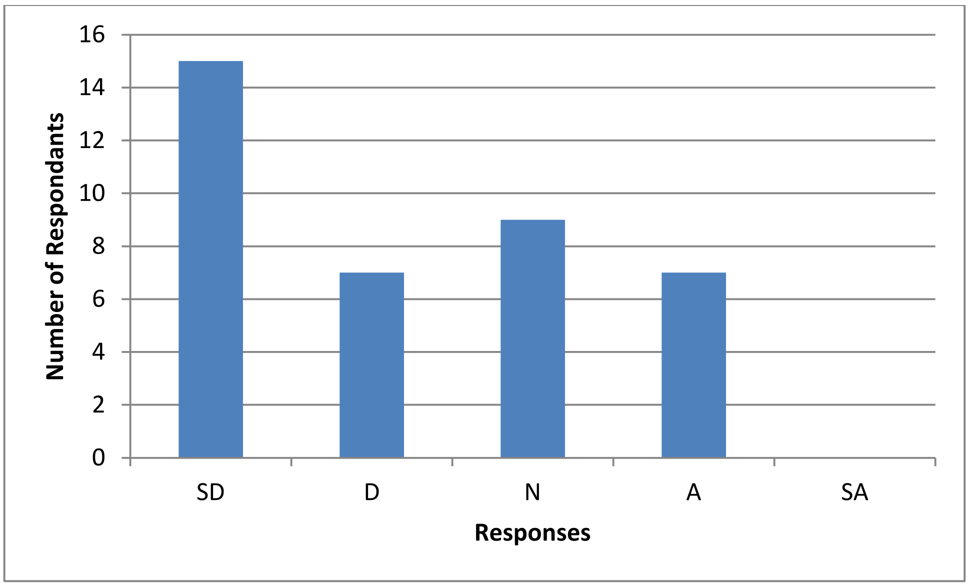


Figure 57 - Distribution of Results for Statement 2

Taken together with statement one (I understand the concept of ‘liking’ and ‘disliking’ changes made to the system as I performed the tasks), this shows that the two framework mechanics for user interaction (providing feedback and permitting reinforcement) have support for their effectiveness. An intrusive feedback mechanic would be both frustrating and lead to frequent interruptions, and so it is important that the framework be as unobtrusive as is possible whilst still supporting the need for providing the information users need to make meaningful decisions.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
18.42%	57.89%	23.68%	SD	Disagree	Disagree	1.22

Table 50 - Post-study Statements 2

11.2.3 Statement Three: I feel it is important that the computer not make changes to my settings without my permission

Previous related research into computer agents have shown that assuming responsibility over a user’s computer creates what is often an unwelcome inversion of control (c.f Trewin, 2000; Rudman and Zajicek, 2006). The ACCESS Framework inverts the responsibility for making corrections, delegating in turn that responsibility to registered plug-ins. It was anticipated that agreement with this statement would be high, consistent with previous research into agent-based configuration. This was supported by the data gathered.

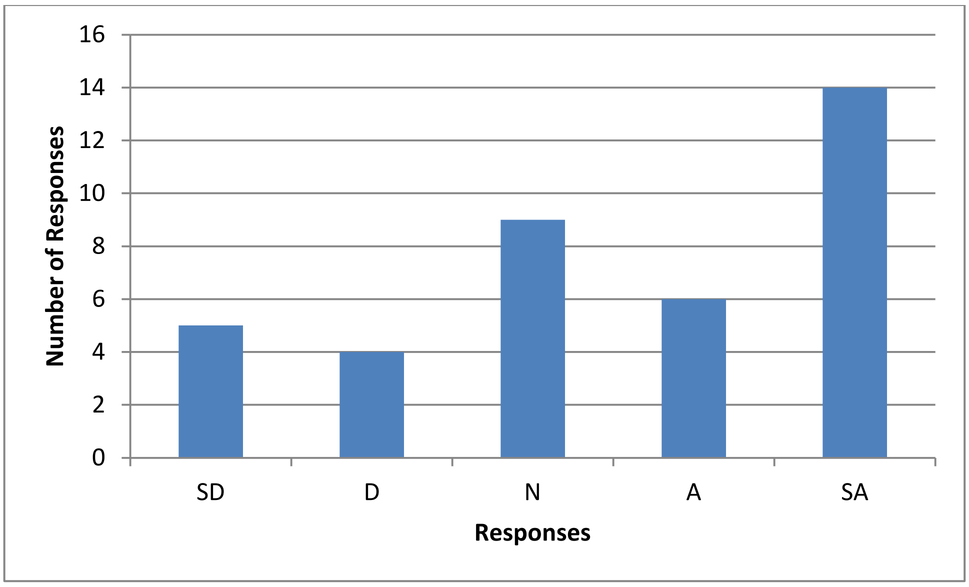


Figure 58 - Distribution of Results for Statement 3

However, it is important to note that this is an expression of a preference without a supporting context. While agreement with this statement is a strike against the fundamental design of the framework, the statement is further specialized later with specific reference to the tool itself.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
52.63%	23.68%	23.68%	SA	Agree	Agree	1.32

Table 51 - Post-study Statements 3

11.2.4 Statement Four: I feel that my permission should be sought before changes are made by the computer

The mechanic employed by the framework is to make a correction and then solicit the user’s opinion. Previous research into configuration agents has suggested that this violation of the internal locus of control implicit in good user interface design is a problem for many users. As such it was anticipated that agreement with this statement would be high, consistent with the literature. This was confirmed by the results gathered:

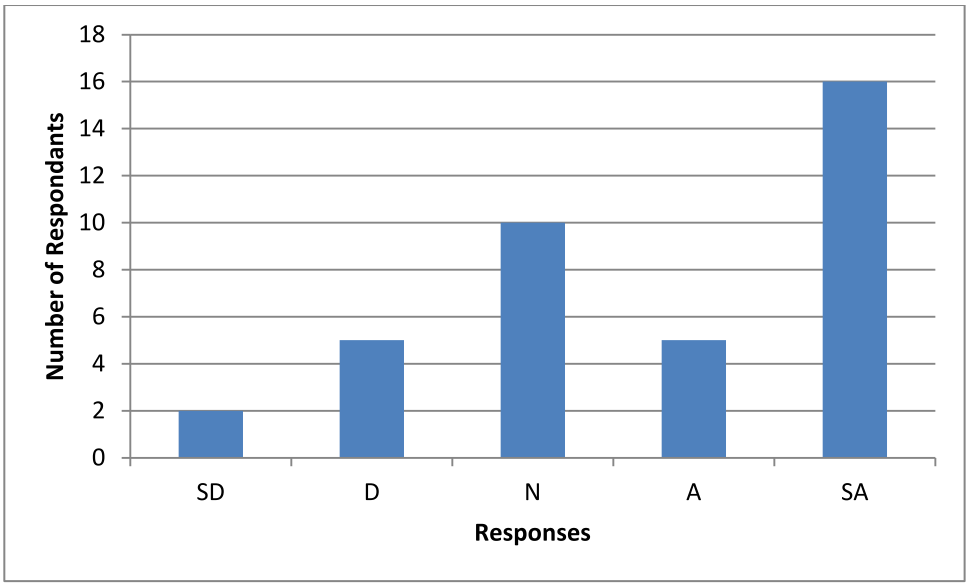


Figure 59 - Distribution of Results for Statement 4

As with statement three (I feel it is important that the computer not make changes to my settings without my permission), this is an indication of a preference without a context, and while on its own it argues against the fundamental design of the framework, it is further specialized in later questions with regards to the operating framework.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
55.26%	18.42%	26.32%	SA	Agree	Agree	1.14

Table 52 - Post-study Statements 4

11.2.5 Statement Five: I would prefer to have individual choice over what settings the computer changes

The way that the framework worked during the study was to pick a plug-in based on the spinning of a weighted roulette wheel. An alternate mechanism could be to present a list of choices for potential corrections and allow the user to pick the one they wish to apply. This technique would not permit for the reinforcement mechanic to adjust relative weightings of plug-ins as cleanly, but could be offered as an alternate mode of configuration. However, there is an anticipated additional burden in this, requiring users to understand the distinction between the different configuration options which would result in greater intrusiveness.

Additionally, there is a danger of simply compounding one of the problems that the framework was designed to resolve, namely that of users lacking the confidence to make changes they know how to make. It was expected that agreement with this statement would nonetheless be high, as was confirmed by the experimental data gathered.

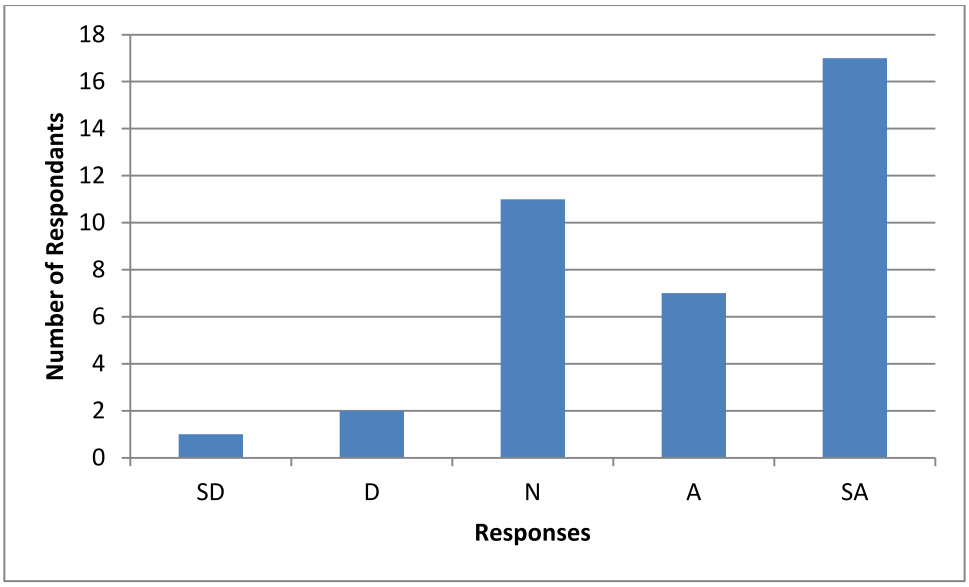


Figure 60 - Distribution of Results for Statement 5

In several situations during the running of the study, users expressed surprise at some of the corrective activities taken by the framework - problems with double-clicking in an earlier task could, if the corrective period overlapped two different tasks, lead to the framework correcting double-clicking when the user's problem was now primarily cursor location. There are several technical solutions to this issue, but the high agreement with this statement argues that a redesign of how plug-ins are presented as correction choices would be popular. However, as outlined above, there are implications to such a redesign that would perhaps reintroduce problems the framework was designed to resolve. This is addressed further in the next chapter of this dissertation.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
63.16%	7.89%	28.95%	SA	Agree	Agree	1.01

Table 53 - Post-study Statements 5

11.2.6 Statement Six: I feel it is okay for the computer to make changes and then solicit my opinion

The selection of plug-ins by the framework does not override user feedback, it inverts it. This statement specialises the general cases above regarding locus of control and focuses it on the mechanism employed by the framework. It was hypothesized that even though individuals would express a preference for more control over the corrections made by the system, that this specific scenario would elicit a favourable response due to the explicit incorporation of user feedback into the statement. However, the results were instead contentious, which does show that while individuals will express a preference for more control, when asked about this specific scenario the feedback is spread evenly between those who agreed and those who did not:

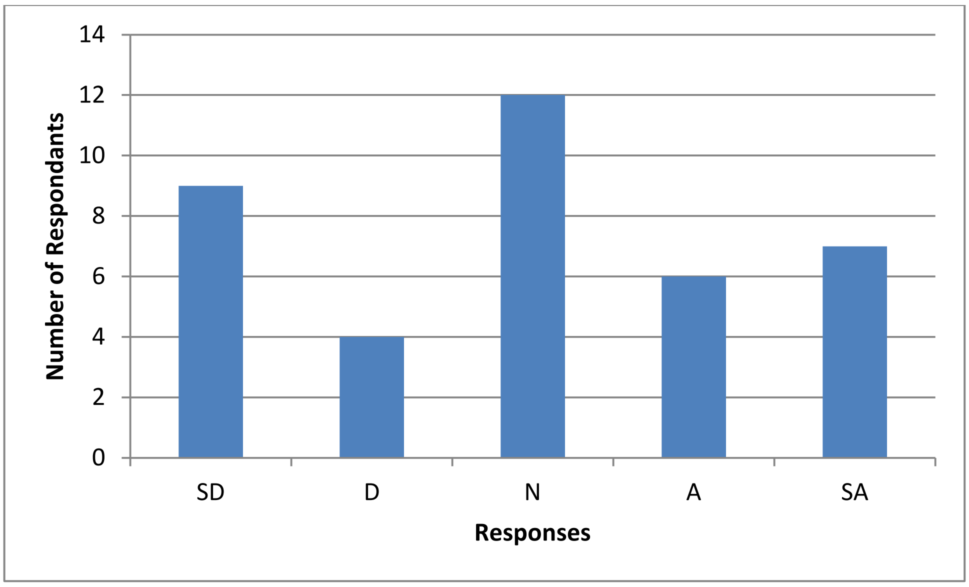


Figure 61 - Distribution of Results for Statement 6

Thus, these results imply that even where a preference for control is expressed, this is contextual and dependent on the exact mechanism used. The results here do not provide evidence for or against the framework, but do show that the general case, when specialized into a more concrete statement, shows the strength of feeling regarding control is malleable.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
34.21%	34.21%	31.58%	N	Agree	Contentious	1.42

Table 54 - Post-study Statements 6

11.2.7 Statement Seven: I am comfortable with the way in which corrections were made

Upon making a correction, the framework would flash up an information dialog complete with the reinforcement options. In cases where the expressed preferences of individuals directly predict comfort with the mechanism, the results from the previous statements regarding user control and configuration should correlate strongly with statements three, four, five and six. These responses would predict a negative response to this statement, but instead the result is more nuanced and leans towards agreement. This suggests that while individuals would express a strong preference on paper, when given an example prototype with which they can become familiar, that the preference does not relate as strongly as might be expected to comfort with the tool.

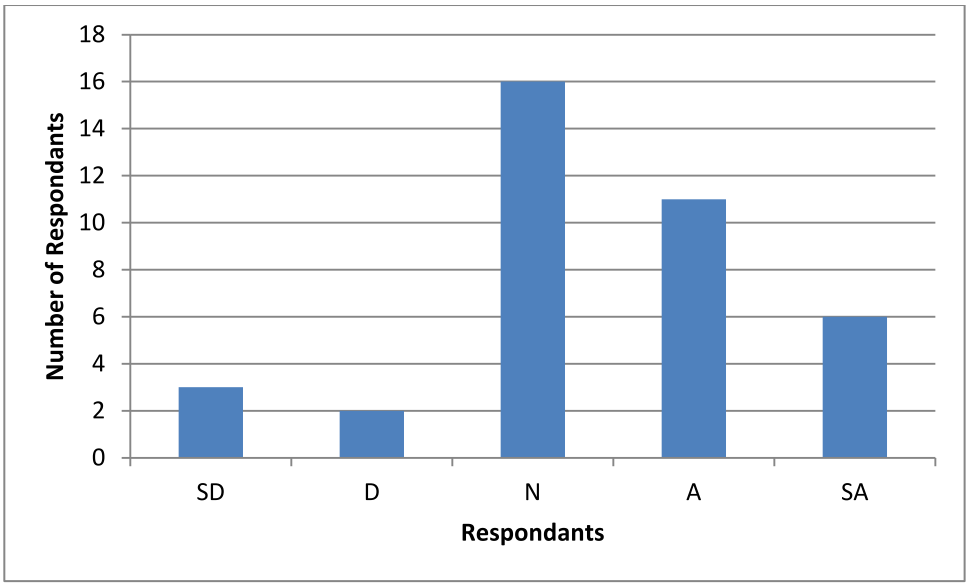


Figure 62 - Distribution of Results for Statement 7

While the agreement rate does not meet the necessary 50% to be considered strong agreement, it is counterbalanced with only slightly more than one in ten of the participants reflecting disagreement. The results thus show support for the design of the framework.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
44.74%	13.16%	42.11%	N	Disagree	Leaning Towards Agree	1.08

Table 55 - Post-study Statements 7

11.2.8 Statement Eight: I am comfortable with the way I was asked for my preferences while performing the tasks

As with statement seven (I am comfortable with the way in which corrections were made), the negative responses to questions three, four, five and six should strongly predict disagreement with this statement – however, the expressed preferences are instead in agreement that they were comfortable with the way the framework asks for their preferences. Once again, this suggests that while an individual may express a strong individual preference for control, their actual hands-on experience with the principle in action leaves a more positive impression:

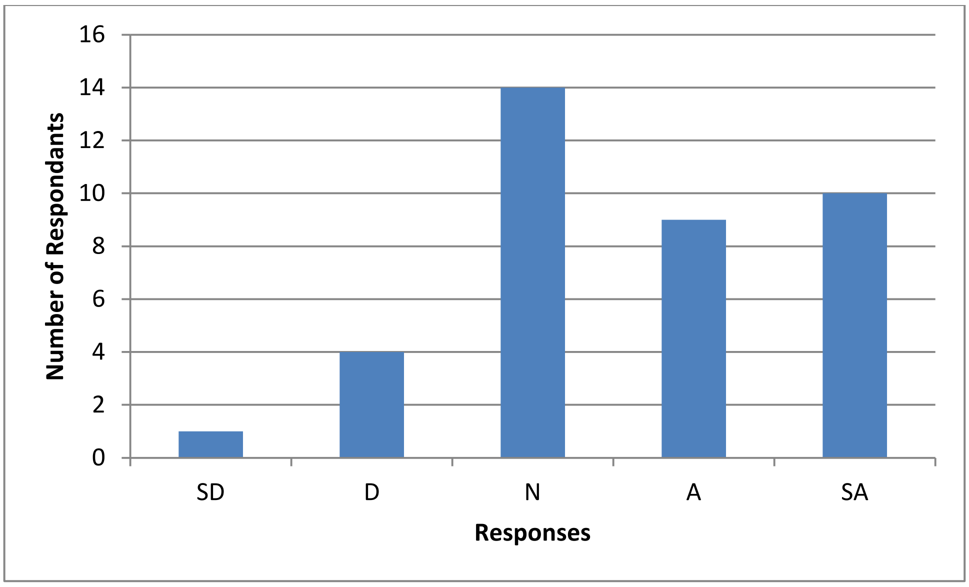


Figure 63 - Distribution of Results for Statement 8

Results for this statement then are strongly supportive of the framework, showing a high level of comfort with the way in which users interact with the system. A high disagreement rate for this statement would invalidate much of the design of the framework, as it is predicated on being a system that is objectively useful, is perceived to be useful, and makes changes in a way that is comfortable for its intended audience. The strong rate of agreement then is important supporting evidence.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
50.00%	13.16%	36.84%	N	Disagree	Agree	1.04

Table 56 - Post-study Statements 8

11.2.9 Statement Nine: I feel that this is an appropriate way of configuring a computer

This statement encapsulates the key deliverable of having an effective, useful framework for user configuration – that of user perceptions of appropriateness. Agreement for statement seven (I am comfortable with the way in which corrections were made) and the relatively high agreement rate for statement six (I feel it is okay for the computer to make changes and then solicit my opinion) would suggest that agreement with this statement would be high, as is supported by the gathered feedback:

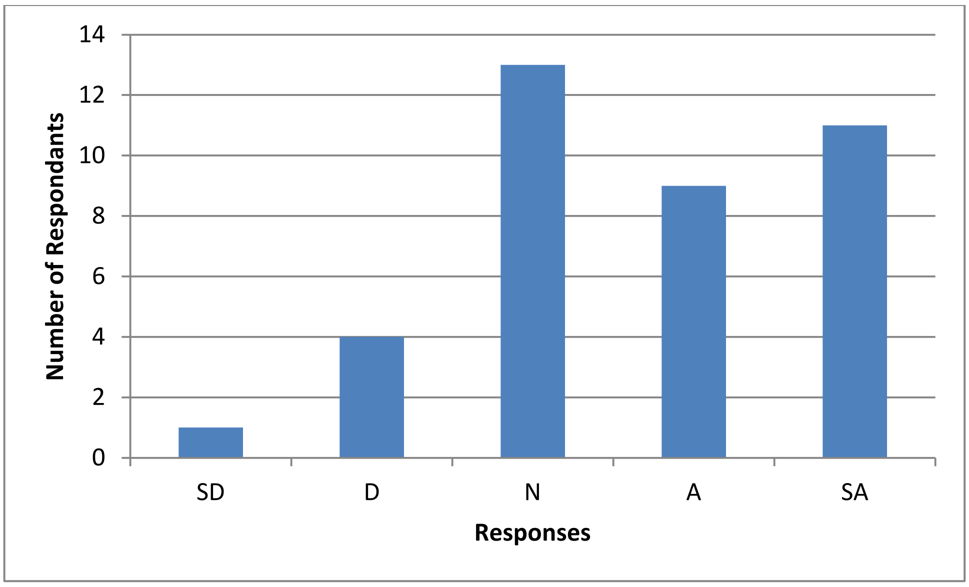


Figure 64 - Distribution of Results for Statement 9

This set of responses indicates again strong support for the design of the framework as presented to participants.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
52.63%	13.16%	34.21%	N	Disagree	Agree	1.10

Table 57 - Post-study Statements 9

11.2.10 Statement Ten: I would be willing to use a system like this on my own computer

Objective and subjective measures of improvement aside, if the framework is not one people would be willing to use on their own systems, it lacks viability as a primary mechanism for configuration support. While the statement is not qualified by issues of cost, privacy, or flexibility of provided plug-ins, support for the framework is high. This is understandable considering the increased subjective values from the experimental condition when the framework is active, and is consistent with the answers given for statements seven, eight and nine.

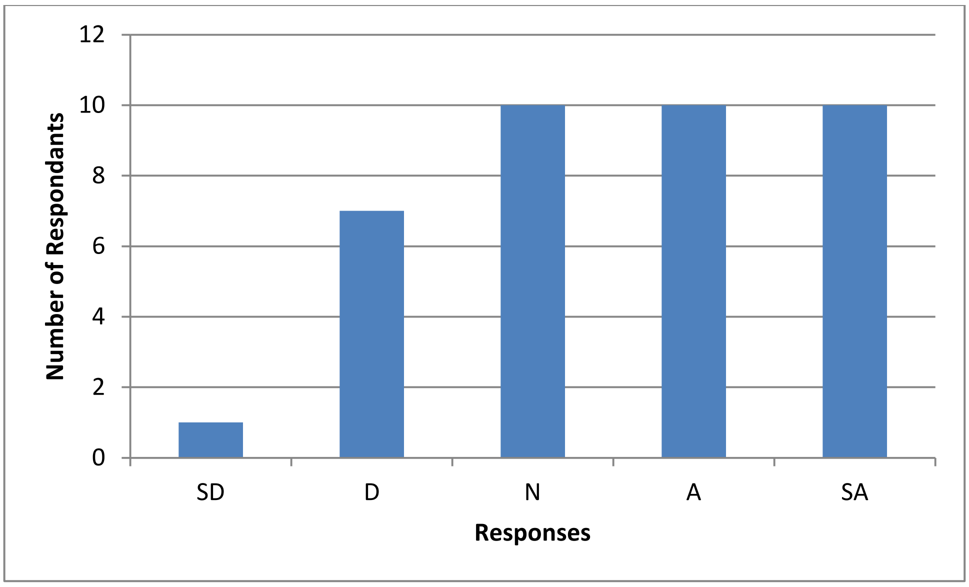


Figure 65 - Distribution of Results for Statement 10

This is strong support for the framework, although it must be qualified by the evidence that older users tend to be more positive about prototypes than is perhaps strictly justified (Eisma et al, 2004). Nonetheless, the results for this statement are positive and this reflects positively on the design of the framework as it currently stands.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
52.63%	21.05%	26.32%	A	Agree	Agree	1.14

Table 58 - Post-study Statements 10

11.2.11 Statement Eleven: I understood the effect of the changes the computer was making

Making genuinely informed decisions about individual preferences with regards to system corrections is predicated on individual users understanding the changes that were being made. It should not be necessary for an individual to understand why a particular correction has been proposed, but being able to indicate a like or dislike requires the user to be able to focus their attention on the changed behaviour of the system.

A special effort is required on the part of plug-in developers to ensure that the language used in corrections is simple, clear, and free of jargon. For the plug-ins assessed as part of this study, the language was chosen to be as straightforward and the changes made to be as visible as was reasonably possible considering their functionality. As such, agreement with this statement was anticipated to be high, which was supported by the gathered results:

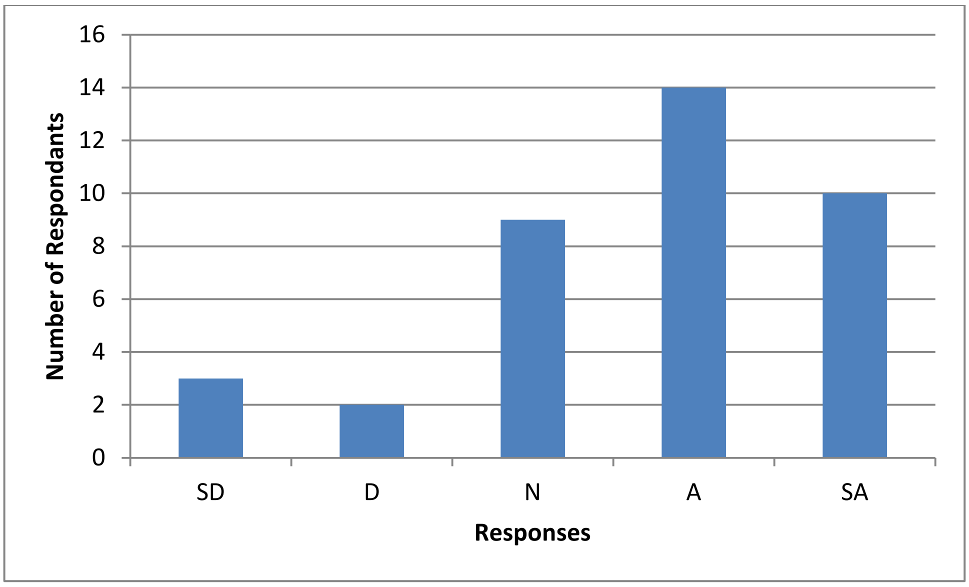


Figure 66 - Distribution of Results for Statement 11

The feedback provided by the framework and the observable impact of changes are the only ways in which users can assess their preferences regarding the corrections. For many configuration changes (double click thresholds, key de-bounce rates and so on), the impact may be invisible and not possible to quickly or easily test. In such cases, the clear language in the description of the change is the only information a user has to act upon. The results for this statement do not justify or invalidate the arguments for the worth of the framework, but do suggest that the language choices in the plug-ins was effectively pitched for the intended audience.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
63.16%	13.16%	23.68%	A	Agree	Agree	1.22

Table 59 - Post-study Statements 11

11.2.12 Statement Twelve: I feel that the changes that the computer made were beneficial

To some extent, the feelings with regards to this statement are already captured in the subjective feedback gathered after each task, but that feedback gives a differential between experimental conditions and does not capture the overall feeling of satisfaction at the end of all tasks. As the subjective feedback shows improvements across almost all tasks, it was anticipated that responses to this statement would show strong support for the changes the framework made, and this is demonstrated in the results gathered:

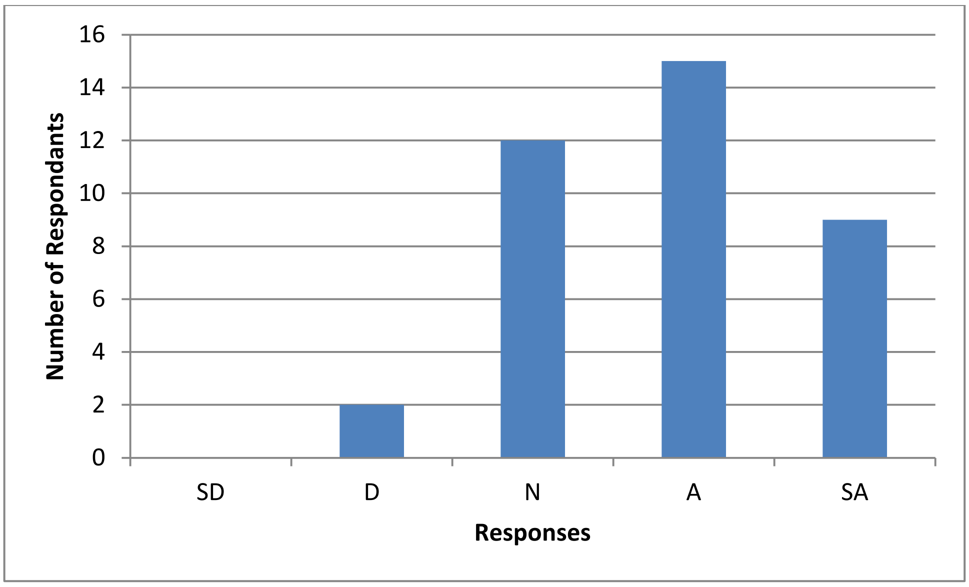


Figure 67 - Distribution of Results for Statement 12

The high level of agreement and low level of disagreement show strong support for the framework, even if the objective results from the trials do not show corresponding increase in performance for two of the tasks. Individuals however clearly believe that the changes made were beneficial - as discussed in the previous chapter, this may be because of some kind of user interface placebo effect. However, two other possible interpretations of the results are that the additional cognitive burden implied by the tool requiring reinforcement could render small improvements in performance difficult to statistically extract. Additionally, the benefit may come from rendering the tasks less frustrating - all three of these interpretations are supported by this gathered data.

Agreement Rate	Disagreement Rate	Neutral Rate	Mode	Expected Result	Actual result	Standard Deviation
63.16%	5.26%	31.58%	A	Agree	Agree	0.90

Table 60 - Post-study Statements 12

11.2.13 Statement Thirteen: I feel that enough information was provided for me to be able to make decisions about my preferences

Related to statement eleven (I understood the effect of the changes the computer was making), this statement specialises those responses and focuses them on the information provided by each plug-in. In an ideal case, the impact of a plug-in's corrective activity is obvious to a user without explanatory text. However, such text must be provided regardless to confirm that a correction has been made and explain its likely impact in layman's terms. Agreement with this statement was predicted to correlate with that in statement eleven, however the result is instead contentious.

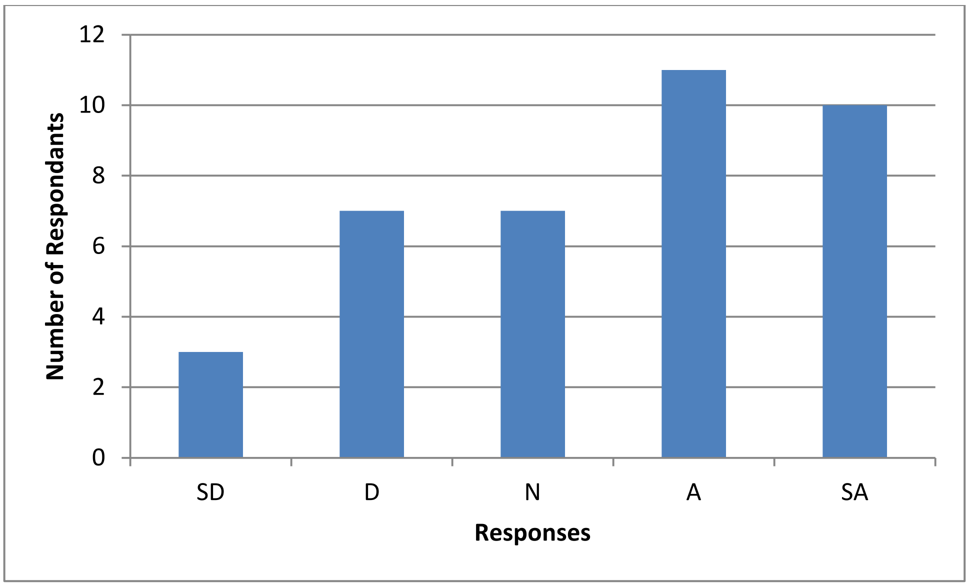


Figure 68 - Distribution of Results for Statement 13

Taking these results along with the results of statement eleven (I understood the effect of the changes the computer was making), it seems to imply that while the language used is reasonably clear, the implication of the user's choices are not properly expressed. As part of the participant briefing, individuals are introduced to the tool, and the impact of their reinforcements are explained. This impact though is never then alluded to again within the running of the tool. This suggests the need for a small tweak to the tool to explain the implications of liking and disliking changes with regards to individual plug-ins.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
55.26%	26.32%	18.42%	A	Agree	Contentious	1.27

Table 61 - Post-study Statements 13

11.2.14 Statement Fourteen: I prefer small changes to be made at a time

It would be entirely within the capability of the framework to make large scale corrections based on analysed data. For the double-click detector, the threshold is adjusted by 100ms at each step. An alternate approach would have been to average out suspected double-clicks and set the threshold to that. However, while this would perhaps result in quicker configuration, it would lose the opportunity for fine-grained adjustments and would make it more difficult for suites of plug-ins to effectively cooperate. The small changes made by the framework permit for incremental assessment of an individual's accessibility profile. There is thus an academic justification for the pace of changes, but this statement assesses whether or not individuals prefer this over larger changes:

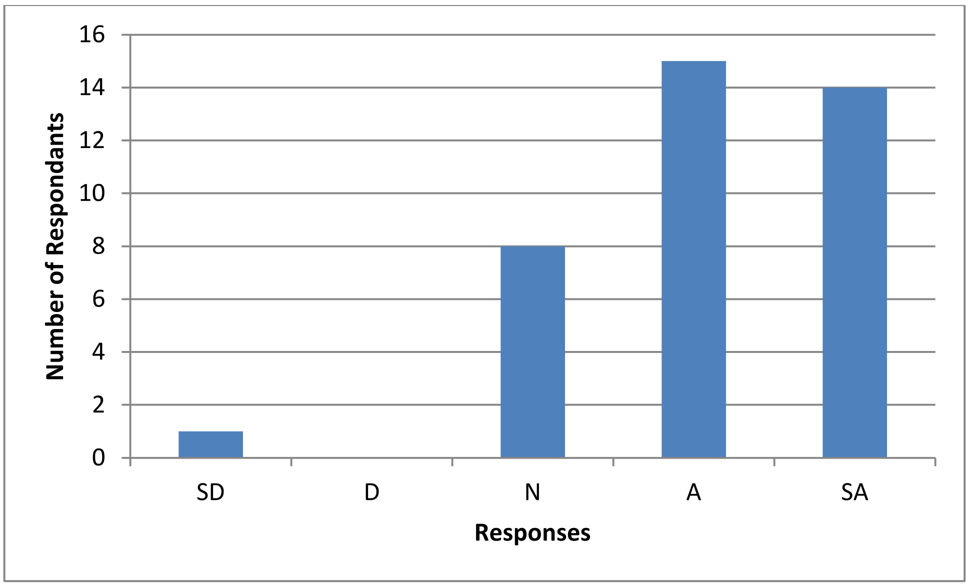


Figure 69 - Distribution of Results for Statement 14

The strong rate of agreement registered by participants show that small adjustments are both academically justifiable, and in line with the preferences of the study participants.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
76.32%	2.63%	21.05%	A	Agree	Agree	0.92

Table 62 - Post-study Statements 14

11.2.15 Statement Fifteen: I felt that enough time was provided in order for me to assess whether or not I liked a change

The pace of adjustments made by the framework in the testing scenario is artificially high. There was a danger that this would create an unfavourable sense of urgency for participants to decide on whether or not the changes they were presented with were useful. A low agreement rate for this statement would have implications for the analysis of both the objective and subjective results gathered through the studies. However, agreement is high suggesting that even with the relatively short amount of time provided for individuals to reinforce, it is still enough to decide whether or not a change is appropriate.

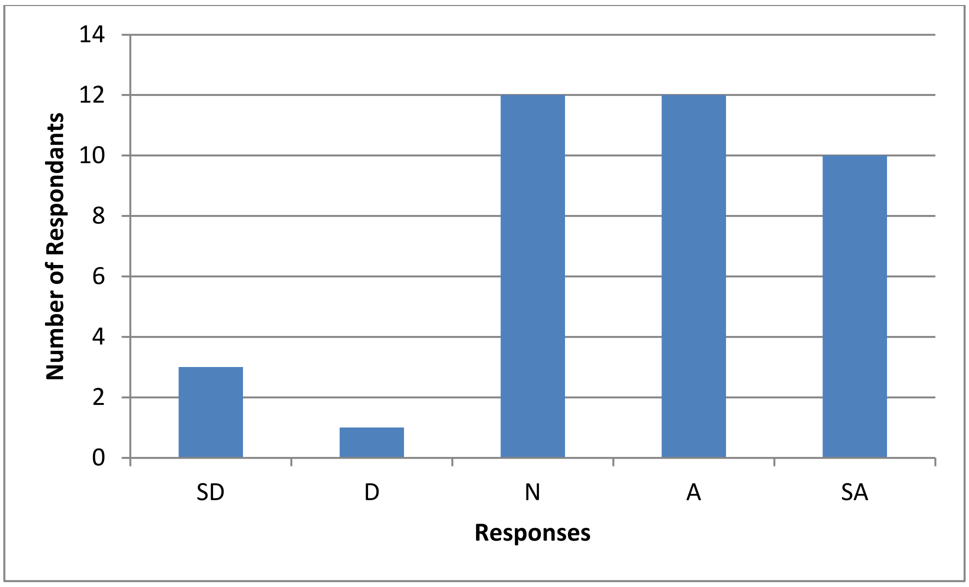


Figure 70 - Distribution of Results for Statement 15

In real world deployments, the time period between ACCESS ticks would be longer than the 60 seconds used in the study, and this would allow for much longer periods of assessment of changes. However, the low rate of disagreement suggests that substantially extending the assessment period is not required, and that the current mechanism of 'small changes made quickly' is appropriate. However, it must be stressed that this is only speculative, and ethnographic testing of the tool would be required to assess appropriate timing periods in in-situ installations.

Agreement Rate	**Disagreement Rate**	**Neutral Rate**	**Mode**	**Expected Result**	**Actual result**	**Standard Deviation**
57.89%	10.53%	31.58%	A/N	Agree	Agree	1.14

Table 63 - Post-study Statements 15

11.3 Discussion of Subjective Results

Subjectively, support for the framework as a primary mechanism for interface configuration is strong. While individuals may express a preference for more control over the way their computers are configured, the feedback given for the framework specifically shows that this preference is not sufficient to impact on the perceived value of the tool within this research context. It likely explains the nuanced results for question seven, but even with this specific statement the agreement rate is almost five times in excess of the disagreement rate.

Key to the findings in this analysis of results are the following:

- The metaphor of training plug-ins through positive reinforcement and positive punishment is appropriate for this group of participants.

- Despite the artificially high rate of corrections within the testing scenario, participants still felt that enough time was provided to make up their minds, and that the system was not intrusive.
- Participants on the whole felt that the tool was appropriate, that its impact was understandable, that the changes it made were beneficial, and that they would be willing to use a system like this on their own computers.

However, contention was observed with regards to how much information the framework provided, and the way in which plug-ins were selected to perform corrections. One possible modification to the framework to resolve the first issue is to have each plug-in formally explain what reinforcement means in the context of their own internals. Currently, when the double-click plug-in is selected to make a correction it displays the following to the user:

> *Double-clicking has been made a little easier to do.*

A balance is required here between the provided information and the relevance to the end-user. While it would be trivial to also include the new and previous thresholds, that information is unlikely to be helpful in allowing someone to make an informed decision. An altered information prompt might however look like this:

> *Double-clicking has been made a little easier to do by increasing the time you can take between one click and another. If you press 'I don't like this', the speed will be set to what it was before and it will be less likely changes like this will be made in the future. If you press 'I like this', more changes like this will be made in the future if they seem worthwhile.*

The more text that is provided in a dialog, the more the user is expected to read and the more difficult it is to extract the key points. Again, a balance must be sought between the provision of necessary information and the brevity of the text. In the next chapter, I discuss a design workshop in which alterations to the framework were proposed, presented and discussed by a randomised subset of study participants. Similarly, adjustments to the framework with regards to how plug-ins were selected and chosen were made and discussed with the group.

11.4 Relation to Hypotheses

In chapter eight, several hypotheses were put forward for each of these studies:

11.4.1 Double Click Study

11.4.1.1 Participants will reinforce the framework

By the criteria outlined chapter seven, this hypothesis is held to be partially confirmed. Non-reinforcement was more common than reinforcement, but an almost equal distribution was observed over the study. The implications of non-reinforcement are discussed further in the next chapter.

11.4.1.2 Participants will rate the task as easier when they are being assisted by the framework

This hypothesis is held to be confirmed on the basis of the experimental evidence gathered in chapter ten. For 'I feel that the task was easy to perform, there is significant improvement recorded (P<0.001, DF=37, T=5.65) when the framework is active, and for the responsiveness of the equipment similarly significant improvements are recorded (P<0.01, DF=37, T=3.16).

11.4.1.3 Participants will complete the tasks quicker when they are being assisted by the framework

This hypothesis is held to be confirmed on the basis of the experimental evidence gathered in chapter ten – the results for the number of clicks (P<0.01, DF=34, T=6.13) and time between clicks (P=0.0375, DF=34, T=1.83) both demonstrate statistically significant improvements.

11.4.1.4 Participants will find double-clicking a moving target to be too difficult in the control session

This hypothesis is held to be confirmed on the basis of the experimental evidence gathered in chapter ten, with an abandonment rate of 100% for the task in both conditions.

11.4.1.5 Participants will find double-clicking a moving target to be difficult but possible in the experimental session

This hypothesis is held to be partially confirmed. While subjective (P<0.05, DF=37. T=1.84 for ease of task only) and objective improvement is recorded when the framework is active (P<0.01, DF=34, T=3.18 for clicks, and P<0.02, DF=34, T=2.95 for speed), the effect is not sufficient to make the task achievable for the majority of participants and the abandonment rate remained 100%. However, the improvement is statistically significant and did result in an almost doubling of individuals who were able to successfully register a double click during the task.

11.4.2 Multiple Plug-in Study

11.4.2.1 Each user will have their own profile of accessibility corrections

This hypothesis is held to be confirmed as per tables 44-48.

11.4.2.2 Each User Will Reinforce Plug-ins differently

This hypothesis is held to be partially confirmed. While some plug-ins were considered generically useful (such as the double-click plug-in), others were held to be almost universally disliked (such as mouse trails). However, between these two extremes there is a difference between how plug-ins are reinforced by individuals, as is evidenced by tables 44-48.

11.4.2.3 Users will positively reinforce more than they positively punish

This hypothesis is held to be confirmed.

11.4.2.4 Users Will Reinforce Plug-ins with Invisible Behaviour

This hypothesis is held to be confirmed.

11.4.2.5 Participants will rate the tasks as easier when they are being assisted by the framework

This hypothesis is held to be confirmed. There is one exception with regards to ease of the task in the double-clicking a moving target task, but statistically significant improvements are observable across the board between the two experimental conditions.

11.4.2.6 Participants will complete the tasks quicker when they are being assisted by the framework

This hypothesis is held to be partially confirmed. For the static double-clicking task, statistically significant improvement is observed in both speed and number of clicks. For the moving target double-clicking task, improvement is for number of clicks only. For the moving buttons task, improvement is recorded in the number of clicks measured only.

11.5 Conclusion

Support for the framework is strong, demonstrating that it is an appropriate mechanism for older users. It is seen as appropriate, beneficial and understandable as well as a piece of software that individuals would choose to install on their own systems. Several statements showed contentious or nuanced results, but the implications of those are discussed in the next chapter in which the results of a collaborative design and development workshop are presented.

Figures gathered regarding reinforcements show that in 84.29% of corrections the correction was committed either as a result of silent or direct consent. In less than 16% of corrections was a plug-in positively punished, and for only 18.4% of the participants was the framework not corrected at all. A limitation of the study is that it does not illuminate the reasons for the high rate of silent consent, and this too is addressed in the next chapter.

12 Framework Follow-up

12.1 Introduction

Several issues remain unresolved from the experimental evidence with regards to the framework. In addition, several participants during the study made suggestions for changes in order to make it more appropriate for the intended audience. These changes could not be folded into the software while the study was running, and once the study had ended there was limited opportunity to evaluate the impact of changes that may have seemed worthwhile on paper but lacking in practice. To address these and related issues, a focus group of previous research participants was convened. As this is a focus group, the feedback can only be considered to be qualitative rather than quantitative, but does serve as a useful basis for analysis.

12.2 Focus Group Design

Participants for the focus group were drawn from the 38 individuals who had gone through the previous study of the framework. Of these 38 individuals, ten participants (N=10, M=6, F=4) were selected for the focus group. Availability of participants and the restricted set of potential candidates influenced the selection, but an external individual handled the recruitment so as to minimise researcher bias. The focus group lasted for three hours, with twenty minutes of comfort breaks built in. The focus group had three main sections, each separated by a comfort break:

1. Discussion of framework issues
2. Collaborative redesign of the tool
3. Open source and the older user

The focus group began with a short introduction to the results of the previous research, and identified several key questions for each of the sessions. Participant comments were recorded in audio format during the focus group and partially transcribed for analysis.

12.3 Session One – Discussion of Framework Issues

12.3.1 Subjective Improvement

The first question addressed was with regards to the subjective improvement that wasn't validated by objective improvement. One participant raised the issue that the improvement was perhaps psychological:

> *P1: If it says that you're going to find something easier, and it doesn't seem easier, then you possibly think it's your fault, and then you're more likely to go along with that.*

P2: I think that's true

M: So you're saying there may be some kind of placebo effect, essentially, of user interfaces?

P3: Yeah

'P1: Yeah, it's psychological.'

Agreement with this statement was high, showing that participants in the study felt that the placebo effect was a viable and perhaps even convincing interpretation of the observed data.

M: When it comes to making corrections, when the software said 'I think this particular thing should be done here', for example – increasing the pointer size, or changing the pointer trails or such. If that didn't seem to be a sensible thing for it to be done in the current situation, do you think that would still lead to this placebo effect?

P4: I would tend to think so, yes. That's my opinion, anyway.

M: When it currently comes up, it says 'You should find double clicking has been easier to do'. Does that make you think then 'Well, if it's not easier that's a problem with me?'

<general murmurs of agreement>

P5: Yes, I think you would.

P7: If it says something like that, should it not psychologically say to your brain 'This is going to be easier', therefore you have an expectation it will be easier'?

M: So if the language was altered so that instead it said 'Double clicking has changed', would that have the same expectation

P6: 'I think that would just make me annoyed'

P4: 'Yeah, you'd see that and you'd be saying 'Well, which way?'

P7: 'In that case you'd actually be expecting a change, even if there was no change. You'd be looking for something that was altered'

No-one in the group identified the possibility of the tool being an extra cognitive burden, and so that possibility was floated by the moderator for discussion:

M: One of the other possible interpretations of the data is – well, they're not actually equivalent tasks. First we're doing without the software, the second we're doing it with the software. When the software is there, there's another thing you have to do, which is you need *to read the text that comes up, and you need to decide whether you liked the change. So there's also the possibility that some of the theoretical improvement might be being swallowed up by the fact you've got another thing you have to do, because you have to switch your attention to the change and then back to the task. Does that seem like a possibility based on your experiences?*

P5: I think that's a possibility yes.

P6: Yes, that's a possibility.

P4: It seems reasonable to think that the message being there distracts you a bit.

P5: Some of the changes were rather small, so it's difficult to decide sometimes if you actually liked it or not.

P3: You didn't really have time though – it came up, and you were off again [with the tasks].

Reception of this idea was lukewarm in comparison to the almost universal agreement that the placebo effect was a suitable explanation for the difference between the two sets of figures. While this only reflects the feeling of the group and does not necessarily translate into an actual explanation, the summarised views of the focus group were that while it was a possibility that distraction and refocusing of attention was an issue, it did not receive the same level of enthusiastic support as the psychological expectation of improvement present in the language of the plug-ins.

12.3.2 Timing of changes

While the results from the study showed strong support for appropriateness of the timing periods, the group were asked about their preference for a slow, gradual series of corrections rather than the rapid, repeated corrections observed during the trial. However, support for this as a change was limited:

P1: That would be too boring

P10: You wouldn't want it dragging out

P9: You want a reasonable time for it to be done, rather than spread over what you're saying.

P6: Plus, if there's a lot of subtle changes there's not really a lot of point in spreading it out anyway.

P8: Certainly for me, I don't spend hours on a computer – I go on a computer for a reason.

M: Would you prefer then if you sat down at the computer and it was more like an eye-test – you do these little tasks and then it tries to figure it out based on that rather than it running constantly in the background and making changes when it thinks it's needed?

P6: It would be quite useful if it came up with something that said 'Would you like to check your settings?', then if you've got the time to do it you can do it and it would be useful from that time on

P8: And then you can go to a thing that says 'remind me later', which you automatically click on.

The timing then of the tool seems to be appropriate as it currently stands, although when the tool presents itself for a correction could be based on a mechanic more subtle than a corrective tick. Three primary possibilities for this:

- Correction as part of an initial test that is conducted at the user's discretion.
    - This would be a variation of the tasks they already performed for the study, and would be more akin to the calibration of a touch-screen.
- Make a list of potential changes and then present them the next time the user starts up the computer
    - The issue of multiple-users on the same system comes into play here
- Have the tool present changes only when it detects the user has gone idle
    - This could be identified by a separate plug-in

Each of these represents a viable mechanic for correction, but further experimental analysis would be required to decide which is optimal. There is no reason that the framework could not support all three of these natively, but the importance of a default interface has already been discussed in the previous chapter. The careless selection of a correction routine could have significant implications for the value of the tool in real world contexts.

12.3.3 Reinforcement Proportion

In over fifty percent of plug-in corrections being made, the framework wasn't reinforced at all. The data gained during the study did not permit for an explanation of this, leading to several possible interpretations. One of the questions in the focus group then sought to obtain some clarity on the reasons for this, as indicated by this extract:

M: In over 50% of the cases, nobody pressed a button. They didn't say 'I like this' and they didn't say 'I don't like this'. Why do you think that might have been?

P3: They weren't sure if they liked it.

P6: I didn’t really read it – I glanced at it and then went ‘Oh, right'

M: When you’re actually focusing on something else, so you’re double-clicking and you’re chasing the thing around and such – how often did you notice that it came up?

P6: I did press it, I know. It’s quite big. But sometimes I don’t even notice things like that if I’m so intent on what I’m doing.

P4: Perhaps it should come with a sound?

P9: Personally, I think that would be annoying. There are already so many beeps and sounds.

P8: Yes, everything in the house already beeps or bleeps

Later in that same discussion:

P5: I think the idea of it just changing things and not telling you would be sort of alright. If it was something to do with your clicking speed or even your cursor. If you didn’t like it, you’d just need to click on it to take it away. It doesn’t need to come up every two minutes, ‘just do it'

M: So, if it makes a change that you don’t like, do you think you would instantly just dismiss it with ‘Oh, I don’t like that.

P5: Oh yes, I’d click on it to reduce it to what it was.

M: I mentioned at the start of the study what happens when you press the button, and then I never mentioned it again – so, do you feel as if you were aware of what the implication of pressing those buttons actually were?

<general murmurs of agreement>

P5: Yes... if you didn’t like it, you pressed the button and it would revert to what it was. If you did like it, you could leave it and it would change it. I think that’s quite obvious.

P4: Yeah, obvious. It’s just a case of whether you noticed it or not.

Later still in the discussion:

M: So would you say then if a change was made you didn’t like, you’d be more likely to notice that change than if it was something you didn’t dislike?

<general murmurs of agreement>

P7: Yes, I think so.

P1: You're always more likely to notice something you don't like.

<general murmurs of agreement>

P8: Yes, I would agree with that.

M: Do you think it would be fair then, and I don't want this to be a leading question, do you think it would be fair to say if 50% of the people didn't actually press any of the buttons, it was because they didn't dislike the change?

<general murmurs of agreement>

P3: Yes, there was just no great impact

P4: You'd be far more likely to press a button if you didn't like it.

P6: It would have more impact if it was not to your liking.

This provides support for the interpretation that when the framework is not positively punished, it is an expression of tacit consent. Changes that are not desirable were considered to be far more jarring than changes that just didn't seem to be making a huge difference. Silent consent was viewed by the majority of the group as the most appropriate way of handling non-reinforcement of the framework, which suggests that no major modifications of the tool are required with regards to this mechanic.

12.4 Session Two: Collaborative Redesign

12.4.1 Locus of Control

Core to the framework is the assumption of responsibility for making changes. In the study, there were several reasons to be cautious of this mechanic of 'change and then consent'. An exchange during the focus group addressed the need for this mechanic:

P4: Why do you need a message at all? If it's doing something that's going to help you?

P5: You'd notice the cursor had changed

P4: That's what I mean – you wouldn't need to be told that it had gotten bigger.

P6: I would prefer 'Would you like the size of the mouse cursor to be increased' rather than 'this has been done'

M: Going back to your point <indicates P4> about why you need a message at all, how would you feel if potentially things that you weren't actually seeing were happening? For example, with the double-click software... right, for double clicking... for the task you were doing, it was set at 200ms. When that was being changed by the software, that value was being changed, so it was going to 300ms and then 400s and 500ms, until you got to a point where it should be comfortable. Other than the point where you notice that it has gotten easier, that's an invisible change, it's just a number changing inside the computer, how would you feel if every now and again it was just changing things without your knowledge, input, or even telling you that it had done it?

<murmurs of general disapproval>

P1: You wouldn't be happy at all

P6: No

P8: You've got to be aware of things... things that are outwith your control.

P5: But you might not even know – you'd just think 'maybe I'm getting better at this'

Later:

P4: If you didn't want it to automatically correct, then you wouldn't set it to let that. Other than that, I don't see the need for a whacking great thing to appear on the screen. They're busy enough screens already without more information. You either employ the utility or you don't.

The general feeling of the group was that it was not appropriate for the software to make changes without informing individuals of the change that was made. To view the impact of reversing the framework assumption and putting responsibility back on the user for selecting corrective activity, participants were shown a variation of the software. In this variation, upon identifying a correction could be made, the framework would give a list of buttons indicating potential corrections, such as 'make the mouse cursor bigger' or 'switch on mouse trails'. Upon pressing a button, the standard user feedback options would appear as before. A screenshot of how this looks is shown in figure 71.

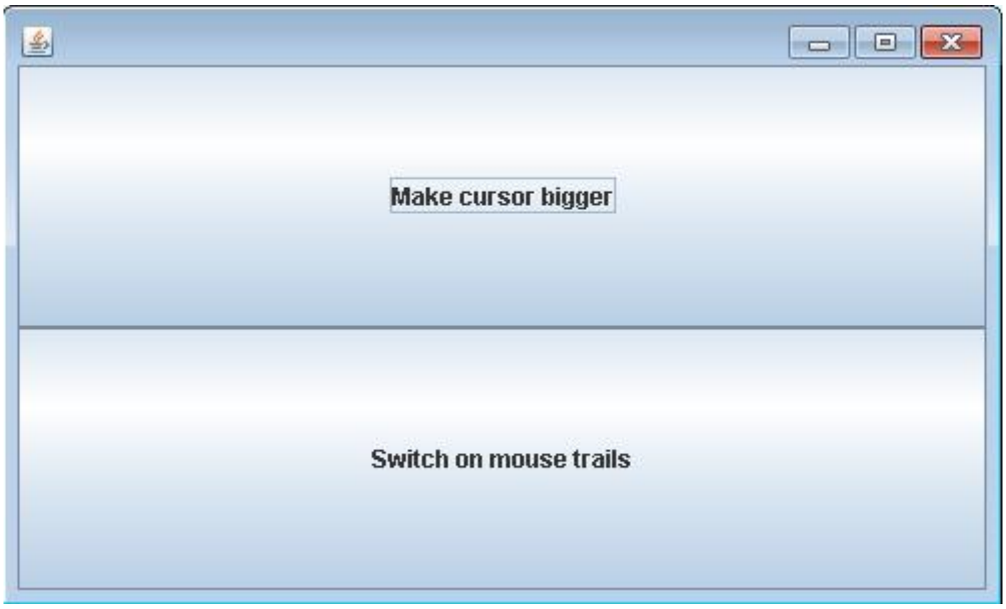


Figure 71 - Suggested Corrections

Assessing this was then passed back to the focus group for discussion:

M: Does that seem like it would be a better way of doing it?

P5: So that might come up two or three times? I think I'd find that quite annoying.

P4: I'd rather if it just made the change and then let me say 'I like this' or 'I don' t like this'. That's me personally though.

P9: I think that would be useful, but I wouldn't want it popping up each time it made a change.

P1: It's just too much to take in.

P4: It's less intrusive the first way.

P5: If you were doing something and that came up, I'd just find it annoying.

M: Can I have a show of hands, please. How many of you would say this way is more intrusive than the first way.

<five hands go up>

M: Given these two different ways of doing it, how many of you would say you prefer this way to the way you experienced it during the study?

<one hand goes up>

Later:

M: So, this doesn't seem like it's any better than before – it seems like it's more intrusive with the buttons needing to be pressed before a change is made.

P4: Yes, less is more.

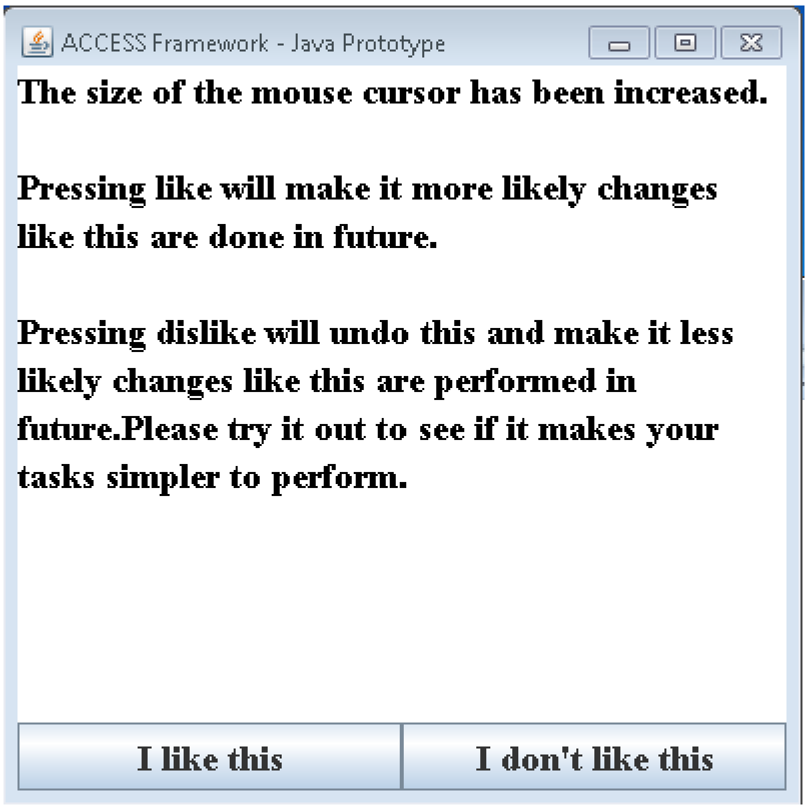
M: Does that seem like a fair summary of what's been said?

<general murmurs of agreement>

The framework in its initial form received the greatest amount of support from the focus group, consistent with the previous interpretation of the data that while individuals may express a preference for control, the mechanic employed by the framework is the least intrusive and most appropriate way of making these corrections.

12.4.2 Information Overload

The answers to the statement 'I feel that enough information was provided for me to be able to make decisions about my preferences' were contentious, and so this was addressed in the study through another variation of the software. In this variation, more information was provided when a plug-in was selected to make a correction, including a description of what would happen when each of the buttons was pressed. An example of the kind of dialog produced is shown in figure 72.



The size of the mouse cursor has been increased.

Pressing like will make it more likely changes like this are done in future.

Pressing dislike will undo this and make it less likely changes like this are performed in future.Please try it out to see if it makes your tasks simpler to perform.

Figure 72 - Full Information on Correction

M: One of the things that came from the study is that, while people understood the liking and the disliking ... there was some suggestion as to what would happen when 'like' or 'dislike' would be pressed wasn't specially clear. What is your view on this as a corrective dialog?

P1: It's too verbose

P8: I think those statements are too pedantic. You're using far too many words, that's my feeling.

P1: Plain English is better.

With less information, as in the study itself:

P6: You read that and you think 'What? Why did you do that? I didn't ask you to do that!'

P5: Usually when those things come up, you can't actually do anything until the button is clicked anyway.

M: Would you assume then when a window like that comes up you have to deal with it right away?

<general murmurs of agreement>

While it was viewed that the full information screen was too wordy, it was also viewed that the stripped down version that they encountered during the trial was not informative enough. Additionally, there was an assumption of modality in the window – that it would not be possible to work in the background while that window was open. Thus, neither of the information implementations of the framework were seen as ideal, and modifications based on this feedback are discussed later in the chapter.

12.4.3 Changes to the Framework

On the basis of feedback received during the focus group, the following changes were suggested as being worthwhile to consider:

- Less leading language
- Options for the order in which corrections are performed
- Additional information provided via options and mouseover text
- Changing the 'I like this' and 'I don't like this' buttons to a green tick and a red cross.
- Buttons on the bottom of the plug-in information window instead of the top
- Positive punishment should remove a plug-in instantly from consideration

Several of these options are uncontroversial. As to the leading language, there is some suggestion in the focus group feedback that a measure of recrimination is associated with someone not identifying an actual benefit. Other feedback suggests that in most cases the impact of changes simply wasn't noticed. Removing the leading language has the potential to nullify the subjective measure of improvement that goes along with plug-ins that have no identified real-world objective improvements. If improvements in self-reported measures of the task were to be matched by corresponding increases in general user satisfaction, then this perhaps would be a feature worth retaining. The study as it currently stands did not provide any data as to whether the significant improvements in subjective feedback translated directly into a feeling of increased satisfaction with the system generally.

Options for changing the order in which corrections are performed are already present in the software as part of the preparation for this focus group. However, the relatively low rate of approval for this workflow suggests that it is not an ideal mechanism. There is though no cost associated with the mechanism being present but set off as default.

Providing additional information via a help button and mouse-over text received generally positive responses, and this information can be dynamically populated as part of the structure of a plug-in. This modification has been incorporated into the framework.

Removing the 'I like this' and 'I don't like this' buttons in favour of a tick and a cross also received generally favourable support – alternate language choices were floated, such as 'continue' and 'delete', or simply 'yes' and 'no', but none of these had the broad approval of simply replacing the contentious language with universally applicable ideogrammatic symbols. Thus, this change too has been incorporated into the framework.

The presence of the buttons at the top of the screen was highlighted as an issue for several participants both in the main study and in the focus group. Moving them to the bottom received enthusiastic support, as this meant that first the information is presented and then as a natural consequence of reading that information the eyes rest on the user choices. This has been incorporated into the framework.

During the study, positive punishment of a plug-in reduced its weighting with regards to it being selected, but popular support was given to the view that positive punishment should stop a plug-in presenting itself ever again. Thus, when a plug-in is positively punished, it is now removed from the potential pool of future plug-ins. Any corrections it has made to date however remain.

12.5 Session Three: Open source and the Older User

The last session of the focus group was a discussion of the issues of open source as it relates to the framework. The assumption of users at the start of the session was that this was software that would potentially be available in the future at a price. However, as discussed in chapter three, this is open source software and the nature of the open source process has implications for tools of this nature. These were addressed in this session of the focus group.

With regards to the cost of software, expense was considered to be a big decision. Most of the participants said they would be willing to try out free software. However, when software asks for personal details, that is a substantial trust issue for users:

> *M: What kind of a decision maker is the cost of software as to whether or not you decide to use something?*
>
> *<general murmurs indicating it is an important indicator>*
>
> *P1: Well, the thing is – you get so much for free. When you go online to get stuff, you find it's all adverts and people trying to get money out of you. There should be a thing that says 'official site', in big letters, and nobody else should be allowed to do that, so that you know that is the actual site.*
>
> *P6: If there was a particular thing you wanted to do with a computer, and you needed the software – maybe it's not a matter of life and death, but it's going to save your bank balance or such, you might invest in it. Otherwise...*
>
> *M: Would you be willing to use a piece of free software just to try it out just because it was free?*
>
> *P6: Oh yes*
>
> *P4: As long as you could trust the source. You can be stuck on the source though. I tried an online credit thing, and all of a sudden it was asking me...*
>
> *P5: Questions you didn't like, yeah*
>
> *P4: And that was... it was supposed to be free.*
>
> *P6: You often go and look something up that's supposed to be free, and you get to a point where it starts to ask you to fill in details, and agree to pay such and such, but you can cancel it within a month.*
>
> *P4: What really cheeses me off about this free software is that you download it thinking it's free, and you find out it's only free for ninety days or so. That's not on.*
>
> *<general murmurs of agreement>*

P6: So I think most of us don't trust those.

<general murmurs of agreement>

Later in that conversation:

M: Is it when it asks you for money that you start distrusting?

<general murmurs of agreement>

M: But if it doesn't ask for money you're quite happy?

P6: No, it's also details.

P4: Yes, as soon as it starts asking for personal details

Branding is considered very important, and participants in the study will look up certifications to make sure they are valid.

P4: As long it's got a brand name, and it's someone you trust. Someone well known, like Microsoft.

P5: If it had a tick from Macafee or Norton saying the thing was clean.

P1: I still wouldn't trust that, because you can still use those names. The only way I'd trust it is if it was something that came through the university, you know, an initiative through the university, and a phone number.

P4: If it has a tick from some of these companies, you can actually go on the company site and see if they're listed.

M: Is that something you would actually do?

P6: Well, I would

P4: Yeah

<general murmurs of agreement>

P4: Yeah, there's just so many viruses floating around now.

Later in the discussion:

P7: Can I say, this is all very interesting all this discussion about brands, but I and I think many people of my age, am a total cop out. I'd ask one of my children.

M: And if they say it's okay you'll be happy enough using the software?

P7: Aye

M: So if you had a piece of software that had, say, the Dundee University logo on it and an IBM logo on it, would that be something that would give you an increased sense of confidence in the software?

P1: Yes

P4: Oh, yes

<general murmurs of agreement>

M: Is it something you would then check and see if it was true? Because anyone can put a logo on anything.

<general murmurs that they would>

Lack of documentation is a problem, but the fact that most software doesn't get packaged with a manual means that seeking online is 'just a thing you have to do'.

M: How important is documentation?

P1: Very

P6: Oh, hugely

<general murmurs of agreement>

M: Would you say having bad documentation is a deal-breaker if it came to using this kind of software?

P6: Well, it's very hard to find documentation.

P10: You could buy it from a source, maybe?

M: What if there's no documentation available at all?

P10 Oh, hrm.

P6: Well, no.

P1: I'd go to a shop, like <name of local store>, to get their advice on it.

Later in the discussion:

M: Are you okay then with using the internet for documentation?

P4: It's just something you've got to do these days.

P6: I think you have to do it, yes.

<general murmurs of agreement>

Thus, some of the initial concerns outlined in chapter three regarding seeking documentation on the internet is simply viewed as a requirement of working with modern software. No-one in the focus group could remember the last time they had received an actual manual with a piece of software, and the PDFs often available on installation discs were widely viewed as inappropriate. However, when shown real world examples of potentially inflammatory language used in response to newcomer requests on forums, most participants felt that it would put them off engaging with such a community. Thus, the often volatile and abrasive nature of open source community boards remains an issue that could result in users disengaging with one of the only means of gaining effective support for an open source tool.

The importance of branding with regards to software is one that is identified in chapter three, and the results of the focus group show that it is an issue raised by the intended audience for the tool. Many participants expressed that they considered the branding and certification important, and that they would not take this certification at face value. This suggests that a suitably official site for the tool could be appropriate provided it could be backed up by official accreditation. All participants expressed distrust of software that did not come with official branding or links to reputable organisations.

Linked to this, it is common for open source software to ask for users to register, and then use their anonymous data as part of the quality assurance process. Most participants indicated that this was a sticking point for them, and that it reduced the confidence they had that free software was actually free. Information gathering with such a tool would be a useful resource and a way to gain valuable information about usage patterns, but this comes at a substantial cost of confidence and it would be recommended that any such information gathering was done purely on an opt-in process through a voluntarily installed plug-in.

12.6 Conclusion

While the results of a qualitative focus group cannot be considered fully representative, and while this particular subset of users represented only a little over 25% of the possible participant pool, a considerable body of feedback was obtained through the three hour session. This information allowed for the framework to be iteratively modified in line with preferences and user feedback. In addition, further information was gathered about the attitude users have to open source software as well as their own interpretations of some of the more unexplained observed data in the study. Specifically, the explanation for the difference between subjective and objective results that had the broadest support was a placebo effect. Additionally, support was strong for the argument that when the framework was not being reinforced, it was likely due to the fact that people either hadn't yet decided, had forgotten they had been asked, or weren’t entirely sure of their preference at the time. This suggests that the approach of silent consent after a number of corrective periods have passed is appropriate. In cases where the tool makes an unfavourable change, the participants in this focus group were quick to insist that they would reinforce to indicate their preferences.

13 Future Work

13.1 Introduction

The ACCESS Framework is aimed at providing benefits at both ends of the accessibility process – for the end users who experience difficulties in configuring their operating systems, and also the developers who are looking to provide accessibility support. The experiments described in this research look at the tool from the perspective of end-users, and this research has yielded promising results. Further work with the framework will include extending this research to add robustness, and other plug-ins aimed at addressing accessibility problems. However, the research to date has not yet investigated the value of the tool to developers, and in this chapter the dissertation will focus on ways in which the usefulness of the tool can be enhanced for deployment, evaluation and refinement with regards to accessibility researchers and accessibility developers generally.

The ad hoc nature of accessibility development for most researchers creates a number of problems with regards to the field going forward on a solid technical base (Oboler, 2003), as well as the comparatively rarefied skill-set associated with effectively architecting a research prototype (Glass, 2003).

Interoperability of research applications is often lacking. Compatibility within families of an operating system, or with different families of operating systems is rarely considered. Extensibility of the software or data is often an afterthought, and the access provided to source code or commercial research data is often restricted by intellectual property obligations (Maurer, 2002; Gambardella and Hall, 2006) as well as a 'result focused' mindset (Oboler, 2003). This can have numerous implications for research (Gambardella and Hall, 2004). Badreddin and Lethbridge (2010) put the problem this way:

> *Research prototypes, particularly those in the software engineering discipline like Umple, face challenges in deployment and validation in industrial practices. Reasons for such challenges may include:*
>
> *1. Research prototypes are frequently not ready for industrial deployment;*
>
> *2. Research tools do not integrate well with other tools, and lack industrial level of reliability and support;*
>
> *3. Risk aversion on the part of industrial users;*
>
> *4. Time constraints and other business commitments;*
>
> *5. Legal regulations or conflict with business objectives*

The framework discussed in this dissertation then, in addition to addressing end-user concerns, is also aimed at resolving issues relevant to those developing this kind of software. There is, as far as this thesis can identify, scant literature available on the subject of developmental issues in accessibility research. Thus the discussion in this chapter is drawn primarily from the author's own observations, the literature of software engineering in general and the literature regarding software engineering in research.

13.2 The Engineering of Accessibility Software

Many of the tasks performed by plug-ins in the ACCESS Framework involve trivial calculations. The coding knowledge necessary to collect and analyse a set of data points and produce a set of outputs is the entirety of the necessary background in software development required to create any of the plug-ins discussed in this dissertation. However, building any of these plug-ins as individual applications greatly increases the burden of knowledge on the software developer due to the increased complexity associated with low level I/O and platform interaction.

Four tasks in particular are performed by all the plug-ins that have been discussed, at least insofar as their deployment on Windows systems is concerned:

- Reading mouse-input from a global system hook
- Reading information from the registry
- Writing information to the registry
- Sending system update calls to refresh loaded configurations

Any developer arriving at the task of creating one of these plug-ins from scratch has a considerable body of work to do before any of the calculations unique to the application can be done. The principle of the ACCESS Framework is 'you should only have to solve the problems that are unique to your plug-in'. The result of this, it is hoped, is a greatly reduced barrier to participation for accessibility researchers from backgrounds that have not stressed software development. It is also hoped that this will result in a generally more productive environment for all accessibility researchers, regardless of technical background.

Research tools are usually written as proof of concept, and as such issues of portability and maintainability are secondary considerations if considered at all. Oboler, Squire and Kobb (2004) have this to say:

> *The inherently high-risk and evolving nature of research renders the risk mitigation approaches of most SDLC[5]s, such as the Spiral, inappropriate. The resultant code is often unusable by anyone but its authors, and, even then, only while fresh in their minds. It is all too often throw-away code, yet research is a continuum. There is a dichotomy between the long-term aims of research and the short-term aims of most research programmers.*

Such software runs the danger of becoming 'abandonware' – the more difficult it is to repurpose a piece of software, then the greater the chance someone will simply rewrite it when the time comes to revisit that functionality (Gardler, 2010). In essence, accessibility researchers are consistently rewriting the wheel – as a rule, we build anew rather than building upon the work others have done. Partially this is undoubtedly due to the difficulty of traversing intellectual property rights, especially in collaborative projects between departments and institutions, but one possible explanation is that it is also simply because there is no culture of reusability amongst accessibility researchers in academia.

When looking at larger issues of development, such as compatibility between and across operating systems, the problem becomes much pronounced. Research tools are mostly written for a specific context, and that context may be as restrictive as 'this computer I have right here'. Porting the software to an updated operating system, or to an entirely different operating system, is a task that must be performed for each individual program. While languages like Java provide a great deal of cross platform support, it is important to acknowledge the limitations of languages that work within a virtual machine – usually, they do not permit access to the low-level input streams required in order to provide accessibility at this level. Languages that do not run in a VM, such as C and C++, usually do have relatively easy access to these streams – but at the cost of portability and the flexibility provided by more modern tools.

All of these problems have at least part of their roots in one core issue – there exists no common framework that actually resolves all of these problems, and because it is not currently perceived to be a major problem in accessibility research (a conclusion which is based on the lack of discussion of the issue in the literature), there is no impetus to provide such a framework. There are reasonable arguments to dismiss each of these problems as relevant to the development of accessibility tool, as is shown in table 64.

[5] An SDLC (Software Development Life Cycle) is a formalised development model for the writing of program code.

Problem	**Argument Against**
It is unreasonably difficult for non-software developers to develop accessibility software.	We can hire a software developer to write the code.
Accessibility software is not maintainable	Research software is proof of concept, and would be rewritten if being used as an end-user tool
Accessibility software is not portable	Most of the users we work with are on Windows systems
Accessibility researchers are often rewriting the wheel	There isn't a common language that provides all the flexibility needed to do anything else

Table 64 - Problems with developing accessibility research software

It is not the thesis' argument that the current situation is one in which productive research is impossible. The argument is that the current situation is one that can be improved with potentially considerable beneficial impact for accessibility researchers generally. The framework described in this dissertation addresses each of these problems, as follows.

13.3 The Barrier of Participation

The design of the ACCESS Framework is to separate out the actionable work of a plug-in from the engine which sets up all of the interactions between the different parts of the system. An accessibility researcher who wants to perform a relatively simple task such as 'keep track of mouse wheel movements, and when the user performs enough rapid reversals of directions, change the speed of the wheel' only needs to understand the role of three classes in the system – the abstract class that defines a plug-in, their own implementation class, and the operating system context which provides the breadth of expression with regards to compensatory activities. In short, the role of their implementation class becomes a mapping between the user input that comes in (as provided by the engine) and the compensatory action they wish to perform when their plug-in is selected.

These plug-ins are written in Java, which is a common language with considerable traction with regards to being considered a good introduction to programming. Most software developers will already know the language, and non software developers will find it more tractable than languages such as C or C++. The amount of understanding required in order to make meaningful developments in this environment is only that of methods, arrays and parameters.

Since the framework provides its own socket architecture, and since the workings of this architecture is abstracted by the engine, even communication between an application and a plug-in can be handled trivially. All that is required is the definition of a method that returns the port on which a plug-in should listen and the handler method that takes incoming information (in string form) and manipulates it. The coding of the application that handles server communications is, alas, outside the scope of the framework to support.

While the task of increasing the expressiveness of the framework is not as simple as creating a single plug-in, it is still something well within the capabilities of the majority of novice programmers. Working within the Windows context for example, there exist methods that permit even a novice to make changes to the registry and refresh those changes across the system. All that a developer needs to know is the name and type of the registry key, and the internal system code to be used to refresh that when it is changed (as is discussed in chapter six).

The additional complexity here comes from the need to ensure extra expressiveness is modelled on all supported contexts. For the purposes of this thesis, only Windows XP and Windows 7 are fully supported, with a partial context available for Ubuntu Linux and another available for Mac OS X. These contexts however lack full expressiveness, but do show that the central concept of a cross-platform framework is sound.

The most experienced of developers would find the lowest levels of the framework – those components that act as the interface between the operating system and the framework itself – to be tractable. It is anticipated that relatively little work would be required within these parts of the framework, as they provide only low level interfaces. Higher levels of the framework successively abstract these low level inputs into a form shared between all plug-ins. In short, it is largely not necessary for anyone to make any changes in these parts of the framework, and certainly not required for accessibility researchers, on a day to day basis, to think about them. Development of these layers would be restricted to the introduction of new input streams, and maintenance of existing code.

13.4 Maintainability of Accessibility Software

Accessibility research is usually goal oriented – the important thing is to try out an idea. Software engineering as a discipline is heavily process oriented: building software in the **right** way is just as important as having correctly functioning outcomes.

Once an accessibility research tool has been developed, unless the results of that tool are formally folded into the portfolio of a spin-off company (Such as with the tool described by Alm, Dye, Gowans, Campbell, Astell and Ellis (2007) which is now a commercial product under the name Circa (http://www.computing.dundee.ac.uk/projects/lim/research.html)), or taken on by a larger corporation with an interest in the field (such as IBM), it usually either gets released to a public repository (OATS), or simply abandoned in the researcher's personal directories. When the source code is available, it is often difficult to adapt, and with only limited support (if any) available from the author.

ACCESS attempts to resolve this problem by creating a single framework that individual accessibility plug-ins can be built upon. Maintenance of particular plug-ins becomes much less important because the aim of the framework is to provide and maintain that common core – the things that change from operating system to operating system are provided externally, and as such even a plug-in that has no support works the way it did (as far as is possible) when it was first released.

However, it should be stressed here that this is an aspirational benefit of the framework rather than one that is guaranteed. It requires that the framework is indeed actively maintained. That one person cannot guarantee this is obvious from the many abandoned projects that exist out in the wild. In this case though, the framework has benefited greatly from the original project brief to create open source software – the entire framework will be released as open source – if a community can be built around it, the project can transcend the limitations of any one developer. Building such a community is considerably outside the scope of this dissertation.

Structuring the framework as it has been done has the potential to greatly increase the lifespan of accessibility tools, as well as to permit fixes in the framework to rattle through the entirety of the available plug-ins. Maintenance as a general rule becomes far less onerous because of the way in which the responsibilities are distributed.

13.5 Accessibility Tools are not portable

It is true that the majority of older users work within Windows operating systems. However, platforms such as the Macintosh have gained market share over the past few years. While such figures are speculative at best and differ from source to source, table 65 (http://marketshare.hitslink.com/os-market-share.aspx) makes available one possible view of the figures. Systems such as Linux remain mostly unknown outside of the technologically adept within this country. However, the result of initiatives such as One Laptop per Child (http://one.laptop.org/) are resulting in much greater market penetration in other countries, especially those countries that are not already possessed of an information technology infrastructure built around Microsoft products.

Date	**Windows Share**	**Mac Share**
August, 2009	93.06%	4.87%
January, 2010	92.21%	5.11%
August, 2010	91.34%	5.00%
January, 2011	89.70%	5.25%
July, 2011	87.60%	5.61%

Table 65 - Market Shares of Windows and Mac

Accessibility tools, tied so tightly to the operating system as they are, are often affected by what might be relatively small changes in design, or names of processes. One simple example is that for the pointer size plug-in, the names of the larger cursors change between Windows XP and Windows 7. That one minor change breaks that plug-in between those two iterations of the operating system unless a conscious effort is made to ensure portability. As has previously been mentioned, languages like Java are cross-platform languages, but in order to perform the tasks that are usually associated with accessibility software, we must move outside the provisions of the virtual machine. Even when we are discussing common functionality, such as running external applications (something supported by the Java virtual machine), trivial differences such as changes in filenames and program location ensure that a conscious effort must be made to ensure compatibility.

An additional implication of this relates to the potential of making use of research tools on the computers of users. Many users have not yet moved on from Windows XP (the W3C statistics tool (http://www.w3schools.com/browsers/browsers_os.asp)) lists the market penetration of Windows XP at 44.2% compared to 32.2% for Windows 7), which means that the developer must be conscious that there may be differences between their operating context and that of their intended deployment audience.

What cannot be guaranteed between platforms is a 100% compatibility as far as end behaviour is concerned. Some operations may take longer on one system versus another, as an example, or the granularity of changes made in one version of Windows may not be supported by another version of Windows. The intention here is not to say that the results derived experimentally from a plug-in deployed on Windows XP will be equivalent to the results derived from the same plug-in running on Ubuntu. Both the nature of the framework contexts and the different designs between the operating systems ensure that plug-ins would need to be experimentally assessed on each individual system.

The framework then provides a consistent interface for developers, rather than a one-to-one equivalency of performance. The corollary to this is that the framework can then only be as expressive as its least expressive context and still maintain functional cross-platform compatibility. While proof of concept work has been done that shows cross-platform compatibility is possible within the framework, there still remains work to do to define the set of common expressiveness that can be shared amongst all contexts.

13.6 Accessibility researchers are often rewriting the wheel

Much developer productivity is lost through writing code from the ground up rather than reusing existing code. This is something that is true across disciplines, and not particularly specific to accessibility. For individuals unfamiliar with the intricacies of event hooks or registry manipulation, there can be frustrating hours spent trying to work out why a particular problem is exhibiting itself. Researching such problems can involve hours of simply staring at the code, internet research, discussions with colleagues and superiors, and so on. At the end of the process, a researcher will end up with a piece of code that does what they need it to do. The contention of this thesis is that there is no need for developers to do this, and it is a waste of developer effort to keep on solving the same low level problems. Developers should be concentrating on the things that make their developments unique.

Deploying a plug-in on the ACCESS Framework is a way in which this can be done. All a plug-in must do is handle the unique functionality that defines the corrective actions and diagnostic routines that you wish to perform. Thus, time spent developing and debugging is spent on the software that achieves the developer's goal, and not the software that must be written before that software can be developed.

13.7 ACCESS and Accessibility Researchers

The discussion outlined above in this chapter is derived from the author's observations, background in software engineering, and the experientially derived principles of software development that have been codified into software development good practice. To support the argument made above, future investigations into the framework should investigate the opportunities to become a genuinely valuable resource for developers as well as for end users.

When developing proof of concept software especially, even experienced software developers make a distinction between the software they are writing to illustrate a point and the software they are writing for end-user consumption. Within the accessibility research community, there is comparatively little emphasis placed on properly building software, with this author being reminded on several occasions that 'nobody gets a doctorate just for writing code'.

While most accessibility researchers have experience with general programming tasks, the nature of programming education means that those coming from non technical backgrounds will find such learning is usually experienced in isolation without a heavy emphasis placed on using such knowledge within real contexts. Additionally, there are issues with the programming courses that are most associated with conversion degrees and introductory programming courses of the kind most experienced by non-software developers moving into accessibility as a research field. These courses tend towards the simpler aspects of programming. Time considerations ensure that deeper contextual information cannot be properly emphasized, and in any case, programming is a skill you build only through practice.

Additionally, the nature of the research process ensures that most accessibility researchers will focus on the end rather than the technical basis of the software they are building. Given a suitably expressive and technically solid foundation, it is one the hypotheses of this thesis that most accessibility researchers will express interest in the availability of a common accessibility framework.

Especially if that framework reduces the amount of programming knowledge required in order to achieve their research ends. However, here it is important to appreciate the differences in motivations people have for developing software. Some researchers (the author included) enjoy software development for its puzzle elements as much as achieving the end goal, and also enjoy the task of programming as an opportunity to refine their own skills. As a corollary then, while a framework with wide developer support will offer benefits and opportunities for experienced developers, it's to be expected that the majority of the benefits of the framework will accrue to novices.

It is position of this thesis, on the basis of the author’s work within the framework, that it greatly simplifies the task of building accessibility research software. Part of the future work associated with this research would be to demonstrate the benefits of this experimentally. As with demonstrating the benefit of the tool from a user's perspective, this falls into three basic categories - identifying a need for the framework, quantifying the benefit, and then assessing the likelihood of adoption.

In the first instance, this would involve thoroughly profiling the existing skill-sets of accessibility researchers, especially those working with older and novice users. It is one of the hypotheses of this thesis that a number of individuals working in this field will hail from backgrounds that are not programming based, and that self-reported measures of confidence with software development techniques and tools will be relatively low. In addition, it is one of the hypotheses of this thesis that accessibility researchers will show a marked emphasis on end-results rather than building a solid technical base for future development.

Quantifying the benefits of the tool is difficult to do from a real world perspective, as many of the possible exercises (such as benchmarking a control group and an experimental group to see how quickly they can develop a specified piece of accessibility software) do not offer a fair comparison. Much of the work needed to build accessibility software is done as part of the framework already, so these types of tasks are not equivalent. However, a profitable avenue for inquiry may be to simply see what relatively novice individuals are capable of within a set period of time using only the framework and a set of accessibility issues.

Finally, assessing the likelihood for adoption is based on accessibility researchers agreeing that there is a need for a common framework, and appreciating the benefit that accrues from its use. Part of this may be simply to offer a larger range of options than are currently supported by the tool and part may be to cultivate an open source community around the tool. The latter of these may be a more convincing demonstration of benefit than any formal experimental study, as the existence of a community of practice around a tool almost by definition, argues for a benefit.

The latter of these in particular is a long-term project that would involve much cultivation of interested researchers, and a constant stream of improvements made to the framework to support the needs of developers working within the field. The expressiveness of the framework as it currently stands is directly related to the expressiveness required to meet the experimental requirements of the research, and this is unlikely to meet the needs of other developers without extension to the current feature-set.

13.8 Extending the Feature Set

There are many valuable options for expanding the feature-set to meet future research requirements. Modern innovations in computer games and technological advances in image capture and analysis have introduced several new potential streams of input that could be supported by the framework. Key amongst these would be eye-tracking (which is now even possible via relatively low end webcams), Wiimotes and the new Microsoft Kinect platform. While APIs exist for these already, a future goal of ACCESS would centre on integrating the interrogation of these devices into a single open source package that allows for plug-ins to pick up data from several devices at once across multiple operating platforms. This would allow for greater opportunities for exploring the interrelation of accessibility issues, such as combining mouse-use and eye-tracking to identify dyslexia or issues of cognitive loading (Chen, Epps, Ruiz, Chen, 2011), or identifying body-language indicators of frustration. It is likely that there is much profitable research to be done in the design space covered by intersecting sets of input.

However, the danger with incorporating additional input streams is in fragmenting the universal applicability of plug-ins which is one of the key future aims of the framework. While much of the user-facing complexity of choosing between plug-ins can be designed away, the implications of this for a tool that might potentially be used on a user's home computer are not well addressed in the literature.

13.9 Ethnographic Satisfaction

The numbers show that in the context of a research setting, the tool seems to be seen as appropriate, useful, and understandable. However, that’s within an artificial, experimental study designed to elicit problems within a limited time scale and this does not show if it’s a tool people could and would use in their homes over a longer period of time. Some of the future work around the framework should focus on an ethnographic deployment of the tool in real world contexts, but there are other issues that need to be investigated and solved as part of this. The most obvious problem is the best way to deliver updated content, bug-fixes, security fixes and new plug-ins. For a younger audience, an online platform similar to Apple's 'App Store' would be appropriate, but the literature on this with regards to older and novice users is non-existent. Before an ethnographic study can be conducted, it is necessary for a technical base of delivery to be implemented, and this involves investigating and assessing a number of deployment vehicles, especially in cases where users may not have easy access to the internet.

13.10 Application Plug-ins

One of the things that emerged from the study was that people generally are just as unwilling to change the settings in their favourite applications as they are changing system settings. At the moment, hooks between plug-ins and applications are very limited (an application can open a socket connection to a plug-in, but there’s no support for that within existing applications). The framework would likely become more useful than current results suggest by permitting easier integration with existing applications. In this way, plug-ins that ‘help you do this thing in Microsoft word’ would be possible. The level of integration permissible from application to application is very likely to be varied, and the differences in application versions are likely to require a more complex engineering solution for plug-ins than is currently permitted by the tool. However, addressing accessibility issues at the application level offers tremendous opportunity for both increasing the viability of the tool and also gaining a wider developer base for plug-ins.

13.11 Cross-System Support

One possibility would also be for the tool to also be cross-system. Individuals would store their accessibility profile remotely, and any time they sat down at a computer they'd be able to bring their profile to the system with them. This would provide support for individuals working within the hot-desk environments that are becoming popular in many organizations. However, discussions with individuals during the experimental studies revealed several reasons to be cautious with this approach - the tactile differences between hardware settings can have a substantial impact on user interaction patterns. What works for one system may not work for another. While it is not technically complex to incorporate such functionality, more research would be required to demonstrate its usefulness.

13.12 Conclusion

The importance of expressing depth rather than breadth of contribution within a PhD dissertation has necessitated a focus on only one of the intended users for the tool, that of the end-user. However, the tool has been written from the start with the intention of it being a useful, beneficial resource for developers. Much of the planned future work around the framework is centred on exploring its value for developers. This involves not only assessing its value, but also extending its current expressiveness so that it can genuinely meet the need of individuals working in areas that may be only tangentially related to that of the author.

Additional input streams, a more effective and cohesive tool-set for cross-platform development, and the creation of a technical architecture for the delivery and deployment of updates and new plug-ins are core to this effort. However, an open source tool is also a useful rallying point for a nascent community of practice. In addition to the technical adjustments to the framework, it's vital to create and cultivate a community around the use and expansion of the tool. There is much work left to be done to make the framework genuinely viable, but the beauty of the open source model is that it is not necessary that the work be done alone.

14 Conclusion

14.1 Introduction

The research questions this thesis was designed to address were discussed in section 1.5. In this chapter, the focus will be on those research questions and what evidence was gathered to support the hypotheses put forward. This chapter then serves to pull together the various pertinent findings and analyse whether or not the ACCESS Framework meets its intended goals.

14.2 Research Problem

Within the introduction, the problems associated with accessibility support for older users were identified as:

- Older users often lack the knowledge of what can be changed within a computer system
- Older users often lack the knowledge to make changes to things they know can be changed
- Older users often lack the confidence to make the changes they know how to make.

For the tool to be generally applicable useful the experimental trials conducted, the participants in the experiments described earlier would have to demonstrate that at least one of these problems hold true for substantial proportions of the group.

These three issues serve as primary motivators for the framework, and have influenced the main elements of its design:

- Users do not need to know what can be changed, because the framework takes a proactive role in identifying accessibility issues.
- Users do not need to know how to make changes, because individual plug-ins have the responsibility for identifying the need for accessibility support and then for enabling or disabling options.
- Users do not need to concern themselves with the problems of undoing changes, because the framework incorporates a simple mechanic for user interaction along with a ‘testing period’ whereby a change can be evaluated.

The key question of the research concerns whether or not this mechanism for user configuration is an appropriate solution to the three issues outlined above. It is the contention of this thesis that the solution would be appropriate if was shown to have a positive impact on user performance, a positive impact on user perceptions of task difficulties, and is seen by users to be a tool with worthwhile properties.

14.3 Analysis of Evidence

14.3.1 The Need for the Tool

The evidence collected from the pre-trial questionnaires for the participants in this study is strongly suggestive that the issues of awareness, knowledge and confidence hold true. Awareness of configuration options for both keyboard and mouse was low, although for mouse options this could be considered contentious from the evidence reported in chapter eight. In either case, there is a substantive subset of the participants who were unaware that settings germane to the tasks were available to be changed. Additionally, the distinction between those changes that operate on an application level versus those that work on a system level was one that a large majority of participants did not appreciate – thus, even in those cases where participants felt that they knew how to make their operating context more appropriate for their own particular needs, it is not clear if they would be able to do that on a system-wide basis as opposed to on an application by application case. Awareness of configuration options thus remains low for the participants in the trial.

Similarly, evidence gathered in chapter eight was strongly suggestive that even in cases where individuals are aware of the changes they can make, they often do not know **how** to make the changes. Even in those cases where the results could be considered contentious, there is a large group of participants who report that they are unaware of how to accomplish many tasks that would improve their computing experience. Configuration options that would have eased certain of the tasks (double click delays, mouse speed and mouse precision) were options of which most individuals felt they would not be able to make use. Similarly for the general case of adjusting mouse and keyboard settings – even in those cases where straightforward agreement (as defined in the chapter) cannot be assumed, there is a substantial number of participants (often a majority) where the issue remains.

Confidence in changing settings is largely predicated on understanding the implications of the change along with the sense that changes can be reversed if they are not appropriate. While most participants felt that they understood the link between their actions and their outcomes (suggesting that they feel they have a working mental model, even if that model may not be accurate), it is still the case that jargon remains an issue and that the consequences of changing unfamiliar settings are not obviously apparent, and that experimentation with a system was a source of discomfort for many participants. The majority of participants simply worked with the system as it was presented to them, rather than attempting to make changes to their operating environment to make it more appropriate.

The consequences of this is that the results from the experiment are likely suitable to be for generalising to a larger audience than that which was exposed to the tool. Applicability of the tool is unlikely to be universal (in general, the more experience individuals felt that they had, the less useful they felt the tool to be), but a substantive proportion of the intended demographic of the tool still demonstrate a need that can potentially by addressed by the ACCESS Framework or another tool of its nature.

14.3.2 The Effectiveness of the Tool

Effectiveness of the framework was assessed in two key areas – objective performance and self-reported, subjective measures of the ease of the tasks. For one plug-in (Double Click), it is possible to specifically link benefit and the plug-in itself. The nature of the Multiple Plug-ins study however is such that it is not possible to indicate a causal link between the presence of a plug-in and objective or subjective measures of improvement. All that can be said for the Multiple Plug-Ins study is that the set of plug-ins as a whole had an impact on the user.

Each of the plug-ins in the study was selected to make corrective actions, and while this does not indicate that it picked up on real accessibility issues, it does demonstrate that plug-ins can co-exist productively and that they can each have an opportunity to make meaningful corrections to a user's system. Additionally, the evidence shows that while a significant number of the changes made by plug-ins went without any kind of user conditioning, it was still the case that the majority of participants did reinforce the framework a number of times. In cases where the framework was left to silently assume consent, participants in the focus group strongly felt that it was as a result of a change being considered harmless, rather than an active decision to cede authority or as a result of unwillingness to register a reinforcement.

As to the impact of plug-ins on tasks, three tasks demonstrated improvements in objective performance in one or more measures. Double-Click is the only one of these that can be singled out as having a direct causal link between plug-in corrections and objective improvements because of its focus within the Double-Click study. As to the other tasks, it is not possible to say which plug-ins contributed to any objective improvements with any degree of certainty. However, for all tasks there are statistically significant improvements in two or more of the subjective measures even in those cases where no objective measure of improvement was recorded.

When there is a good match between plug-in and task, the improvements that were observed from the application of the tool were considerable.

14.3.3 User Ratings of the Tool

The need for a tool coupled with benefits of the tool are of little value if users have no regard for the tool, and as such part of the evaluation of the tool focused on whether users felt that the tool was worthwhile. This was evaluated primarily in the focus group and in the post-trial questionnaire competed by participants. These focused on participant feelings regarding the reinforcement learning mechanism, the tractability of the metaphor, and the level of comfort participants reported.

The results gathered are strongly suggestive that the ACCESS Framework is an appropriate method for configuring user systems. Users felt that the tool was beneficial, un-intrusive, and that it used an understandable metaphor. It was considered to be appropriate, and that the primary mechanism of 'change and then confirm' was also appropriate. Participants felt that the changes the system was making were understandable, and that they would be willing to use a tool like it on their own systems.

These universally positive results should be interpreted in light of the tendency of older users to be unusually positive about software prototypes, but they are regardless suggestive that the framework has real potential to address the issues it was designed to address.

14.4 Limitations of the Research

The nature of the experimental process along with the pressures on the development of research software as discussed in chapter thirteen require a short discussion on the limitations of the framework. Primarily, these lie in the expressiveness of the feature-set of the tool. It is not currently able to override user input, and it is designed explicitly to act as a bridge between existing operating system settings and user input. The latter of these is a direct consequence of the cross-platform nature of the tool. There is nothing that would stop a developer from having a plug-in invoke an external application, but the external application would need to be supported on all platforms if the benefits of cross-platform support were to be maintained.

Additionally, the feature-set at the time of writing is limited to what was required in order to implement the plug-ins for the study. New functionality of this type can be easily added (as discussed in chapter six), but there is work left to be done with regards to extending the feature-set while retaining platform independence.

Within the context of the experimental trials, the tasks undertaken were intentionally designed to provoke user interaction problems rather than model real world interaction behaviour. As such, while the results are promising within the context of the trials, it must be emphasised that this is not the same as the results being applicable within an in-situ context. While it is the position of this thesis that the framework succeeded in answering the research questions, it only answered them in the context of a series of restricted trials with semi-artificial constraints. Individual plug-ins within the framework would still need to be experimentally validated in less clinical settings in order to validate the benefit for users in the real world.

14.5 Table of Research Hypotheses

The following table outlines each of the aspects of the research question from the introduction, and considers them with regards to the evidence outlines in this dissertation:

Research Hypothesis	Sub-Hypotheses	Held to Be
There is a need for a software tool to ease the difficulty of older users with regards to system configuration	Older users often lack awareness of the configuration options available	Confirmed
	Older users often lack the knowledge to change options of which they are aware.	Confirmed
	Older users often lack the confidence to make changes they know how to make	Confirmed
The ACCESS Framework improves objective and subjective measures of performance.	The Double-Click plug-in improves objective measures of performance	Confirmed
	The Double-Click plug-in improves subjective measures of performance	Confirmed
	The suite of plug-ins addressed in the Multiple Plug-Ins Study improves objective measures of performance	Partially Confirmed
	The suite of plug-ins addressed in the Multiple Plug-Ins Study improves subjective measures of performance	Partially Confirmed
The ACCESS Framework is an appropriate mechanism for user configuration	Participants understood the tool	Confirmed
	Participants felt that the tool was useful	Confirmed
	Participants felt that the tool was appropriate	Confirmed

Table 66 - Research Hypotheses

14.6 Conclusion

The results of the experimental studies performed on the ACCESS Framework and its core mechanics suggest that the problem identified in the literature review, even with shifting demographic trends, still remains an issue. While only one plug-in demonstrated unambiguous improvements in objective performance, improvement is shown in almost all the subjective measures of performance captured by the study. Combined with the fact that it has been experimentally shown that it is possible for a plug-in to effectively correct for an identified issue, this makes a case for the value of the framework in resolving user configuration problems. Further investigation, design and development of plug-ins will increase the value as more effective plug-ins are deployed.

All main hypotheses of the study are held to be confirmed by this analysis of result, and this demonstrates considerable support for the framework's design, its potential value for users, and the effectiveness of the reinforcement learning mechanism.

15 References

Aldrick (2009). Dad's Army of experts to rescue banks. [Online] Available from: http://www.telegraph.co.uk/finance/newsbysector/banksandfinance/5111159/Dads-Army-of-experts-to-rescue-banks.html. [Accessed 31st of March, 2011]

Alm, N. (1994). Ethical issues in AAC research. *Methodological Issues in Research in Augmentative and Alternative Communication*, (pp. 98–104).

Alm, N., Dye, R., Gowans, G., Campbell, J., Astell, A., & Ellis, M. (2007). A communication support system for older people with dementia. *Computer*, *40*(5), 35–41.

Alm, N., Gregor, P., & Newell, A. (2002). Older people and information technology are ideal partners. *Proceedings of the International Conference for Universal Design (UD2002).*

Anthony, D., Smith, S., & Williamson, T. (2005). Explaining quality in internet collective goods: Zealots and good samaritans in the case of wikipedia. In *Fall 2005 Innovation & Enterpreneurship Seminar at MIT*.

Australian Bureau of Statistics (2005). Pace of Aging: Australia and Japan. [Online] available from: http://www.abs.gov.au/ausstats/abs@.nsf/bb8db737e2af84b8ca2571780015701e/fcdefb9501c34275ca2571b0000f1a9e!OpenDocument [Accessed 31st of March, 2011]

Backman, L., Small, B. J., Wahlin, A., & Larsson, M. (2000). Cognitive functioning in very old age. *The handbook of aging and cognition 2nd ed*, *2*, 499–558.

Badreddin, O., & Lethbridge, T. C. (2010). A study of applying a research prototype tool in industrial practice. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, FSE '10, (pp. 353–356). New York, NY, USA: ACM.

Bakewell, J (2008). The silent victims of the technology revolution. [Online], available from: http://www.independent.co.uk/opinion/commentators/joan-bakewell/joan-bakewell-the-silent-victims-of-the-technology-revolution-815375.html. [Accessed 31st of March, 2011]

Benkler, Y. (2002). Coase's penguin, or, Linux and the nature of the firm. *The Yale Law Journal*, *112*(2).

Benkler, Y., & Nissenbaum, H. (2006). Commons-based peer production and virtue*. *Journal of Political Philosophy*, *14*(4), 394–419.

Berkowitz, J. P., & Casali, S. P. (1990). Influence of age on the ability to hear telephone ringers of different spectral content. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, (pp. 132–136).

Bezroukov, N. (1999). Open source software development as a special type of academic research (critique of vulgar raymondism). *First Monday*, *4*(10).
URL: http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/696/606

Black, J. B., Carroll, J. M., & McGuigan, S. M. (1986). What kind of minimal instruction manual is the most effective. *SIGCHI Bull.*, *18*, 159–162.

Bonaccorsi, A. (2003). Why open source software can succeed. *Research Policy*, *32*(7), 1243–1258.

Bosman, E. A. (1993). Age-related differences in the motoric aspects of transcription typing skill. *Psychology and aging*, *8*(1), 87–102.

Bothin, A., & Clough, P. (2010). Quantitative analysis of individual differences in Note-Taking and talking behaviour in meetings. In *Proceedings of IADIS Multi Conference on Computer Science and Information Systems*. Freiberg, German.

Brabham, D. C. (2008). Crowdsourcing as a model for problem solving. *Convergence: The International Journal of Research into New Media Technologies*, *14*(1), 75–90.

Bryskine (2011). Hong Kong budget highlights Aging Population Crisis. [Online] Available from: http://www.theepochtimes.com/n2/world/hong-kong-budget-highlights-aging-population-crisis-51817.html. [Accessed 31st of March, 2011]

Butrica, B. A., Schaner, S. G., & Zedlewski, S. R. (2006). Enjoying the golden work years. Tech. rep., The Urban Institute, Washington, DC.

Capitani, E., Sala, S. D., Lucchelli, F., Soave, P., & Spinnler, H. (1988). Perceptual attention in aging and dementia measured by Gottschaldt's hidden figure test. *Journal of Gerontology*. *43*(6), P157–P163.

Carroll, J. M., & Carrithers, C. (1984). Training wheels in a user interface. *Commun. ACM*, *27*(8), 800–806.

Ceaparu, I., Lazar, J., Bessiere, K., Robinson, J., & Shneiderman, B. (2004). Determining causes and severity of end-user frustration. *International Journal of Human-Computer Interaction*. 77(3). 333-356

Centres for Disease Control (2007). The State of Aging and Health in America 2007. [Online]. Available from: http://www.cdc.gov/aging/pdf/saha_2007.pdf. [Accessed 31st of March, 2011]

Cerella, J. (1985). Information processing rates in the elderly. *Psychological Bulletin*, *98*(1), 67–83.

Chadwick-Dias, A., Mcnulty, M., & Tullis, T. (2003). Web usability and age: How design changes can improve performance. *CUU '03: Proceedings of the 2003 conference on Universal usability*, (pp. 30–37).

Chang (2010). Combatting Asia's aging crisis. [Online] Available from: http://blogs.hbr.org/cs/2010/12/combating_the_asian_aging_cri.html. [Accessed 31st of March, 2011]

Chaparro, A., Bohan, M., Fernandez, J., Choi, S., and Kattel, B. (1999). The impact of age on computer input device use: Psychophysical and physiological measures. *International Journal of Industrial Ergonomics*, 24(5):503-513.

Chaput, S., & Proteau, L. (1996). Aging and motor control. *The Journals of Gerontology. Series B*, *51*(6), 346-355

Charness, N., & Bosman, E. (1990). Human factors and design. In J. Birren, & K. Schaie (Eds.) *Handbook of the Psychology of Aging*, (pp. 446–463). San Diego, CA: Academic Press.

Charness, N., Czaja, S., Fisk, A., & Rogers, W. (2001a). Why Gerontechnology? *Gerontechnology*, *1*(2), 85–87.

Charness, N., & Fox, M. C. (2010). *Formal training, older workers, and the IT industry*. In J. McMullin & V. W. Marshall (Eds.). *Aging and working in the new economy: Changing career structures in small IT firms* (pp. 143-162). Williston, VT: Edward Elgar.

Charness, N., & Holley, P. (2001). Human factors and environmental support in Alzheimer's disease. *Aging & Mental Health*, *5*(2 supp 1), 65–73.

Charness, N., Kelley, C. L., Bosman, E. A., & Mottram, M. (2001b). Word-processing training and retraining: effects of adult age, experience, and interface. *Psychology and aging*, *16*(1), 110–127.

Chase, W. (1973). Perception in chess*1. *Cognitive Psychology*, *4*(1), 55–81.

Chen, S., Epps, J., Ruiz, N., & Chen, F. (2011). Eye activity as a measure of human mental effort in HCI. In *Proceedings of the 16th international conference on Intelligent user interfaces*, IUI ’11, (pp. 315–318). New York, NY, USA: ACM.

Chin, J., Fu, W. T., & Kannampallil, T. (2009). Adaptive information search: age-dependent interactions between cognitive profiles and strategies. In *CHI ’09: Proceedings of the 27th International Conference on Human Factors in Computing Systems*, (pp. 1683–1692). New York, NY, USA: ACM.

Chopra, S., & Dexter, S. (2007). Free software and the political philosophy of the cyborg world. *SIGCAS Comput. Soc.*, *37*(2), 41–52.

Cialdini, R. B. (2007). *Influence: The Psychology of Persuasion (Collins Business Essentials)*. Harper Paperbacks, revised ed.

Cohen, G. D. (2007). *The Mature Mind: The Positive Power of the Aging Brain*. Basic Books.

Colazzo, L., Molinari, A., & Tomasini, S. (2008). Is new necessarily good? testing usability of the new office 2007 user interface. In J. Luca, & E. R. Weippl (Eds.) *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications 2008,* (pp. 1371–1379). Vienna, Austria: AACE.

Connelly, & Hasher, L. (1993). Aging and the inhibition of spatial location. *Journal of Experimental Psychology: Human Perception and Performance*, *19*(6), 1238–1250.

Convertino, G., Farooq, U., Rosson, Carroll, & Meyer (2007). Supporting intergenerational groups in computer-supported cooperative work. *Behaviour and Information Technology, 26*(4).

Czaja, S., & Lee, C. (2007a). The impact of aging on access to technology. *Universal Access in the Information Society, 5*(4), 341–349.

Czaja, S. J. (2005). The impact of aging on access to technology. *SIGACCESS Access. Comput.*, (83), 7–11.

Czaja, S. J., & Lee, C. C. (2001). *The Internet and older adults: Design challenges and opportunities*, (pp. 60–78). New York, NY, US: Springer Publishing Co.

Czaja, S. J., & Lee, C. C. (2007b). *Information Technology and Older Adults*. Lawrence Erlbaum & Associates, second ed.

Czaja, S. J., & Sharit, J. (1998). Age differences in attitudes toward computers. *The Journals of Gerontology Series B: Psychological Sciences and Social Sciences*, *53B*(5), P329–P340.

Czaja, S. J., & Sharit, J. (2003). Practically relevant research: capturing real world tasks, environments, and outcomes. *Gerontologist*, *43 Spec No 1*, 9–18.

Davenport, T. H., & Prusak, L. (2000). *Working Knowledge*. Harvard Business Press, 2nd ed.

Delong, D. (2002a). Better practices for retaining organizational knowledge: Lessons from the leading edge. In *Employment Relations Today*. (3), 51-63.

Delong, D. (2006). Living longer, working longer: The changing landscape of the aging workforce: a MetLife study. *MetLife Study, MetLife Mature Market Institute*, Westport, CT. Available from: http://assets.aarp.org/rgcenter/econ/d16687_boomers.pdf [Accessed 11th of August, 2011]

Delong, D. (2007). Searching for the Silver Bullet: Leading Edge Solutions for Leveraging an Aging Workforce. *MetLife Mature Market Institute*. David DeLong & Associates. Concord, MA.

Delong, D. W. (2002b). Uncovering the hidden costs of ’lost knowledge’ in global chemical companies. Available from:

http://www.executiveforum.com/PDFs/DeLong.TheHiddenCosts.pdf [Accessed August 11th, 2011]

Delong, D. W. (2004). *Lost Knowledge : Confronting the Threat of an Aging Workforce*. New York: Oxford University Press.

Dickinson, A., Eisma, R., & Gregor, P. (2002). Challenging interfaces/redesigning users. In *SIGCAPH Comput. Phys. Handicap.*, (73-74), 61–68.

Dickinson, A., Eisma, R., Gregor, P., Syme, A., & Milne, S. (2005a). Strategies for teaching older people to use the world wide web. In *Universal Access in the Information Society*, (4), 3–15.

Dickinson, A., Gregor, P., Mciver, L., Hill, & Milne, S. (2005b). The non browser: helping older novice computer users to access the web. *Electronic Workshops in Computing Series.* 1-6.

Dickinson, A., Newell, A., Smith, M., & Hill, R. (2005c). Introducing the internet to the over-60s: Developing an email system for older novice computer users. *Interacting with Computers*, *16*(7), 621–642.

Dixon, R. A., Kurzman, D., & Friesen, I. C. (1993). Handwriting performance in younger and older adults: Age, familiarity, and practice effects. *Psychology and Aging*, *8*(3), 360–370.

Dohm, A. (2000). Gauging the labor force effects of retiring Baby-Boomers. *Monthly Labor Review Online*, *123*(7).

Eisma, R., Dickinson, A., Goodman, J., Mival, O., Syme, A., & Tiwari, L. (2003). Mutual inspiration in the development of new technology for older people. *Proceedings of Include 2003*. London, pp. 7:252-7:259

Encore (2010). About Encore Careers. [Online], Available from: http://www.encore.org/learn/aboutencorecareers. [Accessed 31st of March, 2011]

Ekerdt, D. J. (2004). Born to retire: The foreshortened life course. *Gerontologist*, *44*(1), 3–9.

Ellis, R. D., & Allaire, J. C. (1999). Modeling computer interest in older adults: The role of age, education, computer knowledge, and computer anxiety. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *41*(3), 345–355.

Ellis, R. D., & Kurniawan, S. H. (2000). Increasing the usability of online information for older users: A case study in participatory design. *International Journal of Human-Computer Interaction*, *12*(2), 263–276.

Engle, R. W., & Bukstel, L. (1978). Memory processes among bridge players of differing expertise. *The American Journal of Psychology*, *91*(4).

Feldman, R. M., & Reger, S. N. (1967). Relations among hearing, reaction time, and age. *J Speech Hear Res*, *10*(3), 479–495.

Fisher, A. (2005). How to battle the coming brain drain. *Fortune*, *151*(6).

Fox, S., (2004). Older Americans and the internet. *Pew Internet & American Life Project.* Available from: http://www.pewinternet.org/Reports/2004/Older-Americans-and-the-Internet.aspx. [Accessed 11th of August, 2011]

Fox, S., (2005). Health Information Online. *Pew Internet & American Life Project*. Available From: http://www.pewinternet.org/PPF/r/156/report_display.asp. [Accessed 11th of August, 2011]

Fozzard, J. (1990). *Vision and hearing in aging*, (pp. 150–170).

Frazier, L., & Hoyer, W. J. (1992). Object recognition by component features: are there age differences. *Experimental aging research*, *18*(1-2), 9–14.

Freudenheim, M. (2005). More help wanted: Older workers please apply. *New York Times*.

Furunes, T., & Mykletun, R. J. (2005). Managers' perceptions of older workers in the hotel and restaurant industry. *International Congress Series*, *1280*, 275–280.

Fox, S (2004). Older Americans and the Internet. *Pew Internet and American Life Project report*. [Online] Available from: http://www.pewinternet.org/Reports/2004/Older-Americans-and-the-Internet.aspx. [Accessed 31st of March,2011]

Gambardella, A., & Hall, B. (2006). Proprietary versus public domain licensing of software and research products. *Research Policy*, *35*(6), 875–892.

Gambardella, A., & Hall, B. H. (2004). Propriety vs. public domain licensing of software and research products. Economics Working Papers ECO2004/15, European University Institute.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design patterns : elements of reusable object-oriented software*. Addison-Wesley, 1 ed.

Gardler, R (2010). Avoiding abandon-ware: Getting to grips with the open development method. [Online], Available from: http://www.oss-watch.ac.uk/resources/odm.xml. [Accessed 31st of March, 2011]

Ghosh, R. A. (2005). Understanding free software developers: Findings from the FLOSS study. In S. Hissam, & K. Lakhani (Eds.) *Perspectives on Free and Open Source Software*. The MIT Press. Cambridge, Mass.: MIT Press, pp. 23–46

Giles, J. (2005). Internet encyclopaedias go head to head. *Nature*, *438*(7070), 900–901.

Giordano, R. (2007). An investigation of the use of a wiki to support knowledge exchange in public health. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work*, (pp. 269–272). New York, NY, USA: ACM.

Glass, R. L. (2003). A big problem in academic software engineering and a potential Outside-the-Box solution. *IEEE Software*, *20*.

Goodman, J., Syme, A., & Eisma, R. (2003). Older adults use of computers: A survey. *BCS HCI 2003*. Bath, UK. *Vol 2.* pp. 25-38

Government Accountability Office (2009). Nuclear Weapons: NNSA and DOD Need to More Effectively Manage the Stockpile Life Extension Program. [Online], Available from: http://www.gao.gov/htext/d09385.html. [Accessed 31st of March, 2011]

Gregor, P., Newell, A. F., & Zajicek, M. (2002). Designing for dynamic diversity: interfaces for older people. In *Assets '02: Proceedings of the fifth international ACM conference on Assistive technologies,* (pp. 151–156). New York, NY, USA: ACM.

Grewal, R., Targonski, B., & Mach, Q. (2008). Minimizing the impact of change on user productivity. In S. Hartmann, X. Zhou, & M. Kirchberg (Eds.) *Web Information Systems Engineering " WISE 2008 Workshops*, vol. 5176 of *Lecture Notes in Computer Science*, chap. 3, (pp. 5–11). Berlin, Heidelberg: Springer Berlin / Heidelberg.

Gurbani, V. K., Garvert, A., & Herbsleb, J. D. (2005). A case study of open source tools and practices in a commercial setting. In *5-WOSSE: Proceedings of the fifth workshop on Open source software engineering*, vol. 30, (pp. 1–6). New York, NY, USA: ACM.

Hambrick, D. (2002). Effects of domain knowledge, working memory capacity, and age on cognitive performance: An investigation of the Knowledge-Is-power hypothesis. *Cognitive Psychology*, *44*(4), 339–387.

Hanson, V. (2001). Web access for elderly citizens. *In Proceedings of the 2001 EC/NSF Workshop on Universal Accessibility of Ubiquitous Computing: Providing for the Elderly (WUAUC'01)*. ACM Press. 14–18

Hanson, V. (2002). Making the web accessible for seniors. *Providing for the Elderly*, *1*(4).

Hanson, V., & Lesser, E. (2009). Implications of an aging workforce: An industry perspective. In S. Czaja, & J. Sherrit (Eds.) *The Future of Work for an Aging Population*,. Johns Hopkins University Press.

Hanson, V. L., Brezin, J. P., Crayne, S., Keates, S., Kjeldsen, R., Richards, J. T., Swart, C., & Trewin, S. (2005). Improving web accessibility through an enhanced open-source browser. *IBM Syst. J.*, *44*(3), 573–588.

Harmon & Markoff (1998). Internal Memo shows Microsoft's Concern over Free Software. [Online], Available from: http://www.nytimes.com/library/tech/98/11/biztech/articles/03memo.html. [Accessed 31st of March, 2011]

Hars, A., & Ou, S. (2001). Working for free? - Motivations of Participating in Open Source Projects. In *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7*. Washington, DC, USA: IEEE Computer Society.

Hart, T., & Chaparro, B. (2004). Evaluation of websites for older adults: How "Senior-Friendly" are they? *Usability News*, *6*(1).

Hawthorn, D. (2003). How universal is good design for older users? In *CUU '03: Proceedings of the 2003 conference on Universal usability*, (pp. 38–45). New York, NY, USA: ACM.

Hawthorn, D. (2005). Training wheels for older users. *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*.

Helve, J., & Krause, U. (1972). The influence of age on performance in the panel d-15 colour vision test. *Acta Ophthalmologica*, *50*(6), 896–900.

Hertel, G. (2003). Motivation of software developers in open source projects: an internet-based survey of contributors to the Linux kernel. *Research Policy, 32*(7), 1159–1177.

Hollender, N., Hofmann, C., Deneke, M., & Schmitz, B. (2010). Integrating cognitive load theory and concepts of humanâ“computer interaction. *Computers in Human Behavior, 26*(6), 1278–1288.

Huey, R. W., Buckley, D. S., & Lerner, N. D. (1994). Audible performance of smoke alarm sounds. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, (pp. 147–151).

Hurst, A., Hudson, S. E., Mankoff, J., & Trewin, S. (2008). Automatically detecting pointing performance. In *Proceedings of the 13th international conference on Intelligent user interfaces*, IUI '08, (pp. 11–19). New York, NY, USA: ACM.

Jagacinski, R. J., Liao, M. J., & Fayyad, E. A. (1995). Generalized slowing in sinusoidal tracking by older adults. *Psychology and aging, 10*(1), 8–19.

Jastrzembski, T., Charness, N., Holley, P., & Feddon, J. (2005). Input devices for web browsing: age and hand effects. *Univers. Access Inf. Soc.*, *4*(1), 39–45.

Jones, S., Fox, S. Generations online in 2009. Available From: *webcite* http://www.pewinternet.org/Reports/2009/Generations-Online-in-2009.aspx. [Accessed 11th of August, 2011]

Johnson, R. E. (1997). Frameworks = (components + patterns). *Commun. ACM*, *40*(10), 39–42.

Johnson, W., Jellinek, H., Klotz, L., Rao, R., & Card, S. K. (1993). Bridging the paper and electronic worlds: the paper user interface. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, (pp. 507–512). New York, NY, USA: ACM.

Jones, L. Y. (1999). *Great Expectations: America and the Baby Boom Generation*. Coward Mc Cann.

Kahneman, D., Knetsch, J. L., & Thaler, R. H. (1991). Anomalies: The endowment effect, loss aversion, and status quo bias. *The Journal of Economic Perspectives*, *5*(1), 193–206.

Keates, S., & Clarkson, J. P. (2002). Countering design exclusion through inclusive design. *SIGCAPH Comput. Phys. Handicap.*, (73-74), 69–76.

Keates, S., & Trewin, S. (2005). Effect of age and parkinson's disease on cursor positioning using a mouse. *Assets '05: Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, (pp. 68–75).

Kelly, B., Sloan, D., Brown, S., Seale, J., Petrie, H., Lauke, P., & Ball, S. (2007). Accessibility 2.0: people, policies and processes. In *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, (pp. 138–147). New York, NY, USA: ACM.

Kirschner, B. (2008). Building a balanced scorecard for open source policy and strategy: a case study of the microsoft experience. In *ICEGOV '08: Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance*, (pp. 226–231). New York, NY, USA: ACM.

Klemmer, S. R., Newman, M. W., Farrell, R., Bilezikjian, M., & Landay, J. A. (2001). The designers' outpost: a tangible interface for collaborative web site. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, (pp. 1–10). New York, NY, USA: ACM.

Kline, D., & Scialfa, C. (1997). *Sensory and perceptual functioning: Basic research and human factors implications*, (pp. 27–54). New York: Academic Press.

Kline, D. W., & Szafran, J. (1975). Age differences in backward monoptic visual noise masking. *Journal of Gerontology*, *30*(3), 307–311.

Knight, J., & Jefsioutine, M. (2002). Relating usability to design practice. In *Proceedings of the 1st European UPA conference on European usability professionals association conference* - *Volume 3*, (pp. 2–12). Swinton, UK, UK: British Computer Society.

Korteling, J. E. (1994). Effects of aging, skill modification, and demand alternation on multiple-task performance. *Human factors*, *36*(1), 27–43.

Kotary, L., & Hoyer, W. J. (1995). Age and the ability to inhibit distractor information in visual selective attention. *Experimental Aging Research: An International Journal Devoted to the Scientific Study of the Aging Process*, *21*(2), 159–171.

Krampe, R. T., & Ericsson, K. A. (1996). Maintaining excellence: deliberate practice and elite performance in young and older pianists. *Journal of experimental psychology. General*, *125*(4), 331–359.

Laberge, J. C., & Scialfa, C. T. (2005). Predictors of web navigation performance in a life span sample of adults. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *47*(2), 289–302.

Lakhani, K. R., & Wolf, R. G. (2003). Why hackers do what they do: Understanding motivation and effort in Free/Open source software projects. *Social Science Research Network Working Paper Series*.

Lave, L. B., Ashworth, M., & Gellings, C. (2007). The aging workforce: Electricity industry challenges and solutions. *The Electricity Journal*, *20*(2), 71–80.

Leahy, D., & Dolan, D. (2009). Digital literacy - is it necessary for eInclusion? In A. Holzinger, & K. Miesenberger (Eds.) *HCI and Usability for e-Inclusion*, vol. 5889 of *Lecture Notes in Computer Science*, chap. 10, (pp. 149–158). Springer Berlin / Heidelberg.

Lee, C. C., Czaja, S. J., & Sharit, J. (2009). Training older workers for Technology-Based employment. *Educational Gerontology*, *35*(1), 15–31.

Lesser, E. (2007). Managing an aging workforce. Available from: http://www.ceoforum.com.au/article-detail.cfm?cid=7992&t=/Eric-Lesser-IBM-Institute-for-Business-Value/Managing-an-aging-workforce [Accessed 11th of August, 2011]

Lesser, E., & Rivera, R. (2006). Closing the generational divide: Shifting workforce demographics and the learning function. International Business Machines (IBM) & American Society of Training and Development (ASTD). Somers: NY, IBM.

Levesque, M. (2004). Fundamental issues with open source software development. *First Monday*, *9*(4).

Levine, J., & Schappert, M. (2005). A mouse adapter for people with hand tremor. *IBM Systems Journal*, *44*(3), 621–628.

Lindholm, T., & Yellin, F. (1999). *Java Virtual Machine Specification*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2nd ed.

Lindqvist, E., & Borell, L. (2010). The match between experienced difficulties in everyday activities after stroke and assistive technology for cognitive support. *Technology and Disability*, *22*(3), 89–98.

Liu, S., & Fidel, R. (2007). Managing aging workforce: filling the gap between what we know and what is in the system. In *ICEGOV '07: Proceedings of the 1st international conference on Theory and practice of electronic governance*, (pp. 121–128). New York, NY, USA: ACM.

Manpower Inc (2010). Knowledge Retention and Transfer in the World of Work. [Online], Available From: http://us.manpower.com/us/en/multimedia/TL-100_KRT%20Position%20Paper_WEB_v2_tcm126-50681_tcm126-50681.pdf. [Accessed 31st of March, 2011]

Mark, G., Gudith, D., & Klocke, U. (2008). The cost of interrupted work: more speed and stress. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, (pp. 107–110). New York, NY, USA: ACM.

Marquie, J. C., Jourdan-Boddaert, L., & Huet, N. (2002). Do older adults underestimate their actual computer knowledge? *Behaviour and Information Technology*, (pp. 273–280).

Maurer, S. (2002). Promoting and disseminating knowledge: the public/private interface. Paper presented to NRC Symposium on Information in the Public Domain. Washington

McGee, D. R., Cohen, P. R., & Wu, L. (2000). Something from nothing: augmenting a paper-based work practice via multimodal interaction. In *Proceedings of DARE 2000 on Designing augmented reality environments*, DARE '00, (pp. 71–80). New York, NY, USA: ACM.

Mcmullin, J. A., & Tomchick, T. (2004). To be employed or not to be employed? : An examination of employment incentives and disincentives for older workers in Canada. In *The Geneva Association. International Association for the Study of Insurance Economics: Etudes et Dossiers No. 285*

McNicoll, T. Aging Crisis will soon hit Developing World. [Online] Available from: http://www.newsweek.com/blogs/wealth-of-nations/2009/09/10/aging-crisis-will-soon-hit-developing-world.html. [Accessed 31st of March, 2011]

Melenhorst, A., & Bouwhuis, D. (2004). When do older adults consider the internet? An exploratory study of benefit perception. *Gerontechnology*, *3*(2), 89–101.

Milne, S., Dickinson, A., Carmichael, A., Sloan, D., Eisma, R., & Gregor, P. (2005). Are guidelines enough? : an introduction to designing web sites accessible to older people. *IBM Syst. J.*, *44*(3), 557–571.

Mistry, P., & Maes, P. (2009). Augmenting sticky notes as an I/O interface. In C. Stephanidis (Ed.) *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments*, vol. 5615 of *Lecture Notes in Computer Science*, chap. 61, (pp.

547–556). Berlin, Heidelberg: Springer Berlin / Heidelberg.

Mitzner, T. L., Boron, J. B., Fausset, C. B., Adams, A. E., Charness, N., Czaja, S. J., Dijkstra, K., Fisk, A. D., Rogers, W. A., & Sharit, J. (2010). Older adults talk technology: Technology usage and attitudes. *Computers in Human Behavior*, *26*(6), 1710–1721.

Monarchy Today (2005). Facts and Figures. [Online], Available from: http://www.royal.gov.uk/HMTheQueen/Queenandanniversarymessages/Factsandfigures.aspx . [Accessed 31st of March, 2011]

Moglen, E. (1999). *Anarchism Triumphant : Free Software and the Death of Copyright*. Centro di studi e ricerche di diritto comparato et straniero.

Moody, G. (2002). *Rebel Code: Linux and the Open Source Revolution*. United States. Perseus Books.

Moore, T. D., & Serva, M. A. (2007). Understanding member motivation for contributing to different types of virtual communities: a proposed framework. In *SIGMIS-CPR '07: Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel doctoral consortium and research conference*, (pp. 153–158). New York, NY, USA: ACM.

Morrell, R. W., Mayhorn, C. B., & Bennett, J. (2000). A survey of world wide web use in Middle-Aged and older adults. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *42*(2), 175–182.

Morris, A., Goodman, J., & Brading, H. (2007). Internet use and non-use: views of older users. *Universal Access in the Information Society*, *6*(1), 43–57.

Nair, S. N., Czaja, S. J., & Sharit, J. (2007). A multilevel modeling approach to examining individual differences in skill acquisition for a computer-based task. *J Gerontol B Psychol Sci Soc Sci*, *62 Spec No 1*, 85–96.

Newell, A., Arnott, J., Carmichael, A., & Morgan, M. (2007). Methodologies for involving older adults in the design process. Proc. *HCI International 2007 Conference*. (pp. 982–989).

Newell, A., Carmichael, A., Gregor, P., & Alm, N. (2003). *Human Factors and Ergonomics*, (pp. 464–481). Lawrence Erlbaum Associates, Inc., First Ed.

Newell, A., & Gregor, P. (2001). User sensitive inclusive design. In *JIM 2001 Interaction Homme/Machine & Assistance*, (pp. 18–20).

Newell, A. F., Dickinson, A., Smith, M. J., & Gregor, P. (2006a). Designing a portal for older users: A case study of an industrial/academic collaboration. *ACM Trans. Comput.-Hum. Interact.*, *13*(3), 347–375.

Newell, A. F., Gregor, P., & Alm, N. (2006b). HCI for older and disabled people in the Queen Mother Research Centre at Dundee University, Scotland. In *CHI '06 extended abstracts on Human factors in computing systems*, CHI EA '06, (pp. 299–302). New York, NY, USA: ACM.

Nichols, D. M., & Twidale, M. B. (2002). Usability and open source software. *First Monday*, *8*.

Nielsen, J (2005). Scrolling and Scrollbars. [Online] Available from: http://www.useit.com/alertbox/20050711.html. [Accessed 31st of March, 2011]

Nielsen, J (2005). Middle-Aged Users' Declining Web Performance. [Online], Available From: http://www.useit.com/alertbox/middle-aged-users.html. [Accessed 31st of March, 2011]

Norman, D. A. (1988). *The Psychology Of Everyday Things*. Basic Books.

Notess, M., & Lorenzen-Huber, L. (2007). Online Learning for Seniors: Barriers and Opportunities. *eLearn*, *2007*(5).

Oboler, A. (2003). Examining the use of software engineering by computer science researchers. In *Proceedings of Education Students' Third Regional Research Conference*, (pp. 37–45).

Oboler, A., Squire, D., & Korb, K. (2004). Software engineering for computer science research - facilitating improved research outcomes. *International Journal of Computer and Information Science*, (pp. 24–34).

Office of National Statistics (2009). Population Estimates for the UK, England, Wales, Scotland and Northern Ireland [online]. Available from: http://www.statistics.gov.uk/statbase/Product.asp?vlnk=15106 [Accessed 31st of March, 2011]

Oreg, S., & Nov, O. (2008). Exploring motivations for contributing to open source initiatives: The roles of contribution context and personal values. *Computers in Human Behavior*.

Orlikowski, W. J. (1992). Learning from notes: organizational issues in groupware implementation. In *CSCW '92: Proceedings of the 1992 ACM conference on Computer-supported cooperative work*, (pp. 362–369). New York, NY, USA: ACM.

Owsley, C., Sekuler, R., & Siemsen, D. (1983). Contrast sensitivity throughout adulthood. *Vision research*, *23*(7), 689–699.

Paulson, L. D. (2001). Mainframes, Cobol still popular. *IT Professional*, *3*(5), 12–14.

Petch, G. (1998). The cost of lost knowledge. *Knowledge Management Magazine*.

Peterson, P. G. (1999). *Gray Dawn*. Crown, 1st ed. New York: Times Books

Peterson, S. J., & Spiker, B. K. (2005). Establishing the positive contributory value of older workers:: A positive psychology perspective. *Organizational Dynamics*, *34*(2), 153–167.

Pilz, M. (2009). Initial vocational training from a company perspective: a comparison of british and german In-House training cultures. *Vocations and Learning*, *2*(1), 57–74.

Pitt-Catsouphes, M., & Smyer, M. A. (2005a). Businesses: How are they preparing for the aging workforce? (Issue Brief No. 02). Chestnut Hill, MA: Boston College Center on Aging & Work/Workplace Flexibility. Available from http://agingandwork.bc.edu/documents/IB02_BusinessPreparing.pdf [Accessed 11th of August, 2011]

Pitt-Catsouphes, M., & Smyer, M. A. (2005b). Older workers: What keeps them working? (Issue Brief No. 01). Chestnut Hill, MA: Boston College Center on Aging & Work/Workplace Flexibility. Available from http://www.bc.edu/content/dam/files/research_sites/agingandwork/pdf/publications/IB01_OlderWrkrs.pdf [Accessed 11th of August, 2011]

Pitt-Catsouphes, M., & Smyer, M. A. (2007). *The 21st century Multi-Generational workplace*. (Issue Brief No. 09). Chestnut Hill, MA: Boston College Center on Aging & Work/Workplace Flexibility. Available from http://agingandwork.bc.edu/documents/IB09_MultiGenWorkplace.pdf [Accessed 11th of August, 2011]

Platman, K., & Taylor, P. (2004). Workforce ageing in the new economy: A comparative study of information technology employment. Available from http://www.wane.ca/PDF/Platman&TaylorSummaryReport2004.pdf [Accessed 11th of August, 2011]

Plude, D. J., & Hoyer, W. J. (1981). Adult age differences in visual search as a function of stimulus mapping and processing load. *Journal of gerontology*, *36*(5), 598–604. URL http://view.ncbi.nlm.nih.gov/pubmed/7264245

Price Waterhouse Cooper (2010). PwC says reset retirement timetable in response to fiscal and social changes. [Online] Available from: http://www.ukmediacentre.pwc.com/content/Detail.aspx?ReleaseID=3594&NewsAreaID=2. [Accessed 31st of March, 2011]

Pullin, G., & Newell, A. (2007). Focussing on Extra-Ordinary users. In *Universal Acess in Human Computer Interaction. Coping with Diversity*, (pp. 253–262).

Rabbitt, P. (1993). Does it all go together when it goes? the nineteenth bartlett memorial lecture. *The Quarterly journal of experimental psychology. A, Human experimental psychology*, *46*(3), 385–434.

Raymond, E. S. (1998). *Homesteading the noosphere*. In *First Monday*, 3(10). http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/621/542

Raymond, E. S. (1999). *The magic cauldron*. In *ALS'99: Proceedings of the 3rd annual conference on Atlanta Linux Showcase*, (p. 25). Berkeley, CA, USA: USENIX Association.

Raymond, E. S. (2001). *The Cathedral & the Bazaar (paperback)*. O'Reilly. URL http://www.worldcat.org/isbn/0596001088

Reagle, J. (2007). Do as I do: authorial leadership in wikipedia. In *WikiSym '07: Proceedings of the 2007 international symposium on Wikis*, (pp. 143–156). New York, NY, USA: ACM.

Richards, J. T., & Hanson, V. L. (2004). Web accessibility: a broader view. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, (pp. 72–79). New York, NY, USA: ACM.

Salthouse, T. A. (1984). Effects of age and skill in typing. *Journal of experimental psychology. General*, *113*(3), 345–371.

Salthouse, T. A., & Babcock, R. L. (1991). Decomposing adult age differences in working memory. *Developmental Psychology*, *27*(5), 763–776.

Salthouse, T. A., & Prill, K. A. (1988). Effects of aging on perceptual closure. *The American journal of psychology*, *101*(2), 217–238.

Sargeant, M. (2006). The employment equality (age) regulations 2006: A legitimisation of age discrimination in employment. *Industrial Law Journal*, *35*(3), 209–227.

Sayago, S. and Blat, J. (2009). About the relevance of accessibility barriers in the everyday interactions of older people with the web. In *Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A)*, W4A '09, pages 104-113, New York, NY, USA. ACM.

Scotland's Futures Forum (2007). Growing Older and Wiser Together. [Online] Available from: http://www.scotlandfutureforum.org/assets/files/Growing%20Older%20and%20Wiser%20-%20Forum%20report.pdf [Accessed 31st of March, 2011]

Sharit, J., Hernández, M. A., Czaja, S. J., & Pirolli, P. (2008a). Investigating the roles of knowledge and cognitive abilities in older adult information seeking on the web. *ACM Trans. Comput.-Hum. Interact.*, *15*(1), 1–25.

Sharit, J., Hernández, M. A., Czaja, S. J., & Pirolli, P. (2008b). Investigating the roles of knowledge and cognitive abilities in older adult information seeking on the web. *ACM Trans. Comput.-Hum. Interact.*, *15*(1), 1–25.

Shaw, G., & Smith, D. (2003). Don't let knowledge and experience fly away: Leveraging scarce capability to support ongoing competitiveness in the aerospace & defense industry. Available from http://www.accenture.com/SiteCollectionDocuments/PDF/ad_workforce.pdf [Accessed 9th of October, 2011]

Shen, C., Pitt-Catsouphes, M., & Smyer, M. A. (2007). Today's Multi-Generational workforce: A proposition of value. (Issue Brief No. 10). Chestnut Hill, MA: Boston College Center on Aging & Work/Workplace Flexibility. Available from http://agingandwork.bc.edu/documents/IB10_MultiGenValue.pdf [Accessed 11th of August, 2011]

Schieber, F. (2006). Vision and aging. In J. Birren, & K. Schaie (Eds.) *Handbook of the Psychology of Aging* , (pp. 129-154). Academic Press, 6th ed.

Shirky, C. (2008). *Here Comes Everybody*. Allan Lane: New York.

Shneiderman, B. (1997). *Designing the User Interface*. Addison Wesley, 3rd edition.

Shneiderman, B. (2002). Promoting universal usability with multi-layer interface design. *SIGCAPH Comput. Phys. Handicap.*, (73-74), 1–8.

Singh, V., & Twidale, M. B. (2008). The confusion of crowds: non-dyadic help interactions. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*, CSCW '08, (pp. 699–702). New York, NY, USA: ACM.

Sokoler, T., & Svensson, M. (2007). Embracing ambiguity in the design of non-stigmatizing digital technology for social interaction among senior citizens. (pp. 297–307).

Sowe, S., Stamelos, I., & Angelis, L. (2008). Understanding knowledge sharing activities in free/open source software projects: An empirical study. *Journal of Systems and Software*, *81*(3), 431–446.

Staggers, N., & Norcio, A. F. (1993). Mental models: concepts for human-computer interaction research. *Int. J. Man-Mach. Stud.*, *38*, 587–605.

Statistics Bureau (2008). Population by Age. [Online] Available from: http://www.stat.go.jp/english/data/kokusei/2005/kihon1/00/02.htm [Accessed 31st of March, 2011)

Statistics Canada (2006). Portrait of the Canadian Population in 2006. [Online] Available from: http://www12.statcan.ca/census-recensement/2006/as-sa/97-551/p2-eng.cfm [Accessed 31st of March, 2011]

Stobie, K. (2005). Too darned big to test. *Queue*, *3*(1), 30–37.

Streb, C. K., Voelpel, S. C., & Leibold, M. (2008). Managing the aging workforce: Status quo and implications for the advancement of theory and practice. *European Management Journal*, *26*(1), 1–10.

Stroop, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, *18*(6), 643–662.

Sunstein, T. A. (2008). *Nudge: Improving decisions about health, wealth and happiness*. Penguin.

Sweet, R. E. (1985). The mesa programming environment. In *Proceedings of the ACM SIGPLAN 85 symposium on Language issues in programming environments*, (pp. 216–229). New York, NY, USA: ACM Press.

Syme, A., Dickinson, A., Eisma, R., & Gregor, P. (2003). Looking for help? supporting older adults' use of computer systems. *Proceedings of IFIP INTERACT03: Human-Computer Interaction 2003*, (p. 924).

Szyperski, C. (2002). *Component Software: Beyond Object-Oriented Programming (2nd Edition).* Addison-Wesley Professional, 2 ed.

Tapscott, D., & Williams, A. D. (2006). *Wikinomics: How Mass Collaboration Changes Everything*. Portfolio Hardcover.

Thatcher, J (2003). Web Accessibility – What Not To Do. [Online]. Available from: http://www.jimthatcher.com/whatnot.htm. [Accessed on 31st of March, 2011]

Thevenin, D., & Coutaz, J. (1999). Plasticity of user interfaces: Framework and research agenda. In *Proceedings of Interact99.* Interact '99, (pp. 110-117). Edinburgh.

Thornton, R, & Light, L. (2006). Language Comprehension and Production in Normal Aging. In J. Birren, & K. Schaie (Eds.) *Handbook of the Psychology of Aging* , (pp. 129-154). Academic Press, 6th ed.

Tirole, J., & Lerner, J. (2000). The simple economics of open source. *Social Science Research Network Working Paper Series*.

Todman, J., & Drysdale, E. (2004). Effects of qualitative differences in initial and subsequent computer experience on computer anxiety. *Computers in Human Behavior*, *20*(5), 581–590.

Thovtrup, H., & Nielsen, J. (1991). Assessing the usability of a user interface standard. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology* , CHI '91, (pp. 335-341). New York, NY, USA: ACM.

Trewin, S. (2000). Configuration agents, control and privacy. In *CUU '00: Proceedings on the 2000 conference on Universal Usability*, (pp. 9–16). New York, NY, USA: ACM.

Trewin, S. (2002). An invisible keyguard. In *Assets '02: Proceedings of the fifth international ACM conference on Assistive technologies*, (pp. 143–149). New York, NY, USA: ACM.

Trewin, S. (2004). Automating accessibility: the dynamic keyboard. In *Assets '04: Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*, (pp. 71–78). New York, NY, USA: ACM.

Trewin, S., Keates, S., & Moffatt, K. (2006). Developing steady clicks:: a method of cursor assistance for people with motor impairments. *Assets '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, (pp. 26–33).

United Nations (2002). World Population Aging, 1950-2050. [Online] Available from: http://www.un.org/esa/population/publications/worldageing19502050/ [Accessed 31st of March, 2011]

United Nations (2008). World Population Prospects, the 2008 Revision. [Online] Available from: http://www.un.org/esa/population/publications/wpp2008/wpp2008_highlights.pdf. [Accessed 31st of March, 2011]

U.S. Census Bureau (2005). 65+ in the United States. [Online] Available From: http://www.census.gov/prod/2006pubs/p23-209.pdf [Accessed 31st March, 2011]

Vanderheiden, G. C. (2008). Ubiquitous accessibility, common technology core, and micro assistive technology: Commentary on "computers and people with disabilities". *ACM Trans. Access. Comput.*, *1*(2), 1–7.

Vassileva, J. (1996). A Task-Centered approach for user modeling in a hypermedia office documentation system. User Modeling and User-Adapted Interactions.,*6*(2), 185-223

von Hippel, E. (2001). Innovation by user communities: Learning from open-source software. *Sloan Management Review*, (Summer), 82–86.

Von Krogh, G. (2003). Open-Source software development. *MIT Sloan Management Review*, *44*(3), 14–18.

Vouligny, L., & Robert, J. M. (2005). Online help system design based on the situated action theory. In *CLIHC '05: Proceedings of the 2005 Latin American conference on Human-*

computer interaction, (pp. 64–75). New York, NY, USA: ACM.

Walker, N., Philbin, D. A., & Fisk, A. D. (1997). Age-related differences in movement control: adjusting submovement structure to optimize performance. *The journals of gerontology. Series B, Psychological sciences and social sciences, 52*(1).

Wasko, M., & Faraj, S. (2000). â it is what one doesâ: why people participate and help others in electronic communities of practice. *The Journal of Strategic Information Systems, 9*(2-3), 155–173.

Whittaker, S., Swanson, J., Kucan, J., & Sidner, C. (1997). TeleNotes: managing lightweight interactions in the desktop. *ACM Trans. Comput.-Hum. Interact., 4*, 137–168.

Wright, P., Belt, S., & John, C. (2003). Fancy graphics can deter older users: a comparison of two interfaces for exploring healthy lifestyle options. In E. O'neill, P. Palanque, & P. Johnson (Eds.) *Proceedings of HCI 2003 Designing for Society*, (pp. 315–325).

Wu, C. G., Gerlach, J. H., & Young, C. E. (2007). An empirical analysis of open source software developers' motivations and continuance intentions. *Inf. Manage., 44*(3), 253–262.

Wyatt, S. (2003). How users and non-users matter. In N. E. J. Oudshoorn, & T. J. Pinch (Eds.) *How Users Matter. The Co-construction of Technologies and Users*, (pp. 4–22). Cambridge, Massachusetts: MIT Press.

Yan, J. H., Rountree, S., Massman, P., Doody, R. S., & Li, H. (2008). Alzheimer's disease and mild cognitive impairment deteriorate fine movement control. *Journal of Psychiatric Research, 42*(14), 1203–1212.

Zajicek, M. (2003). Software design for older adults to support memory loss. *INCLUDE 2003 Inclusive design. for society and business*. London, UK. Helen Hamlyn Centre. 25-28 March 2003

Zajicek, M. (2004a). Passing on good practice: Interface design for older users. In *Lecture Notes in Computer Science, 1301*, 636–640. Germany: Springer-Verlag

Zajicek, M. (2004b). Successful and available: interface design exemplars for older users. *Interacting with Computers*, *16*(3), 411–430.

Zajicek, M. (2007). Web 2.0: hype or happiness? In *W4A '07: Proceedings of the 2007 international cross-disciplinary conference on Web accessibility (W4A)*, (pp. 35–39). New York, NY, USA: ACM.

Zaphiris, P., Ghiawadwala, M., & Mughal, S. (2005). Age-centered research-based web design guidelines. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, (pp. 1897–1900). New York, NY, USA: ACM Press.

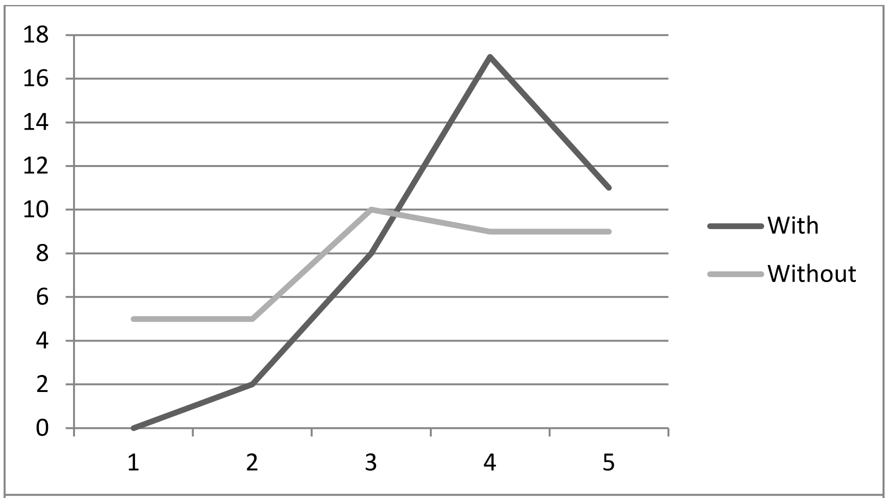
Zaphiris, P., & Sarwar, R. (2006). Trends, similarities, and differences in the usage of teen and senior public online newsgroups. *ACM Trans. Comput.-Hum. Interact.*, *13*(3), 403–422.

16 Appendices

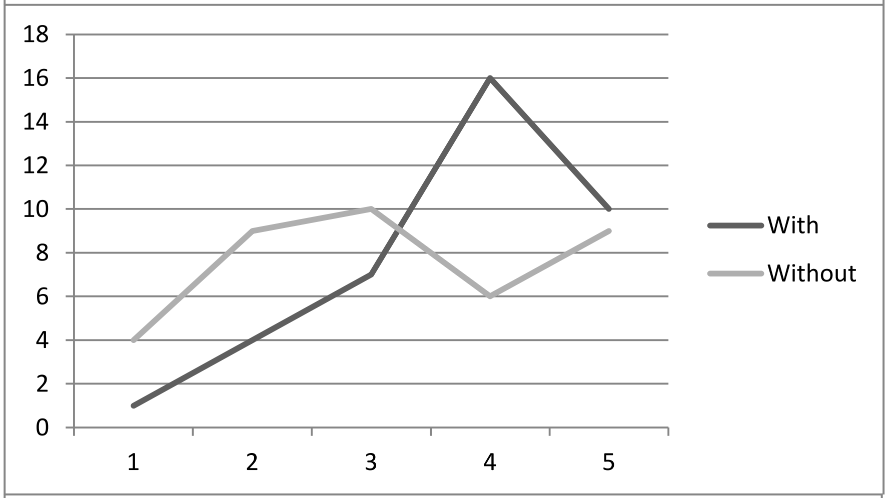
1	Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task One
2	Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Two
3	Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Three
4	Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Four
5	Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Five
6	Tasks for Participants in the Electronic Notes study
7	Instructions for Participants in the Electronic Notes study
8	Code for the Double-Click Plug-in
9	Base for Pointer Size and Mouse Trails Plug-in
10	Code for the Mouse Trails Plug-in

16.1 Appendix One – Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task One

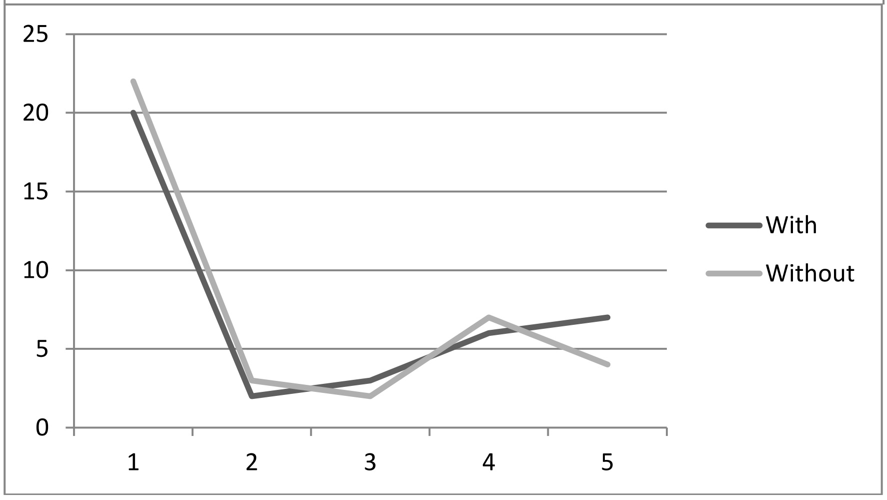
I feel that the task was easy to perform



I feel that the mouse was suitably responsive for the task

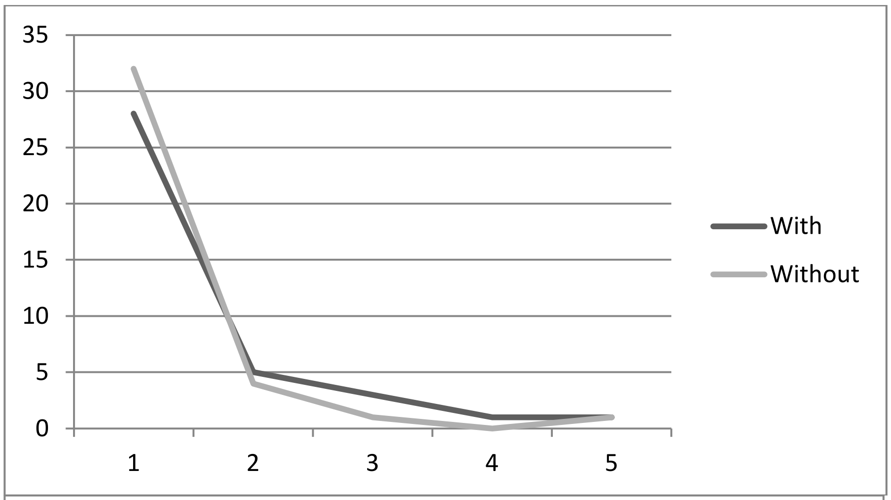


I know how to change the computer settings so that the mouse was more suited to perform the task

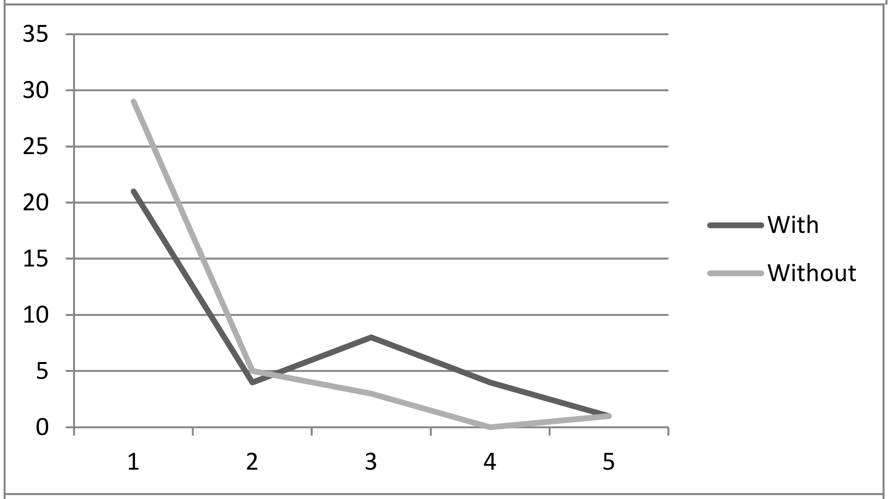


16.2 Appendix Two – Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Two

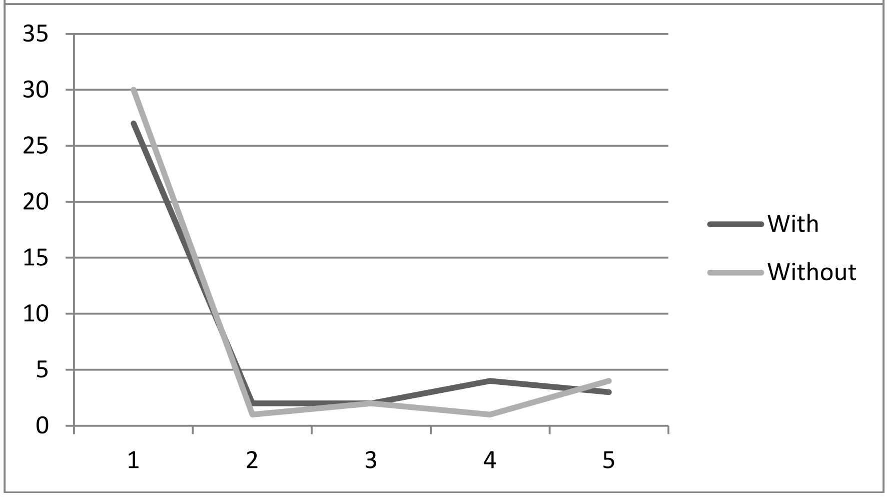
I feel that the task was easy to perform



I feel that the mouse was suitably responsive for the task

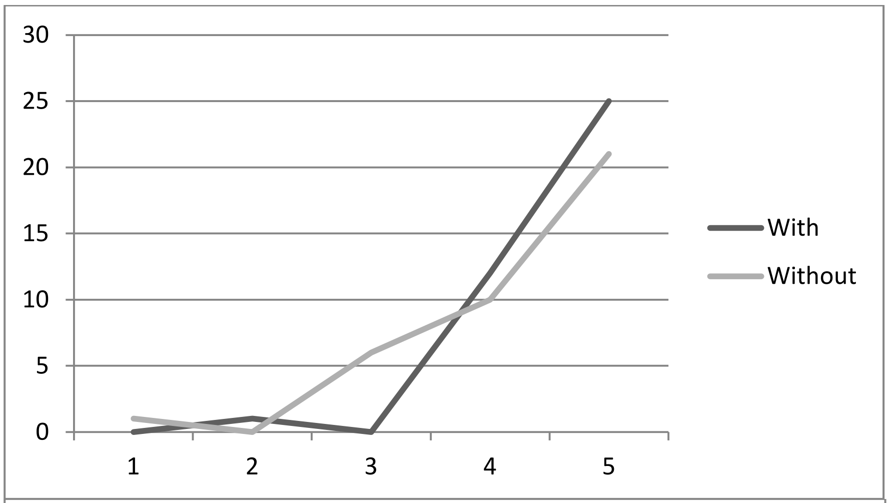


I know how to change the computer settings so that the mouse was more suited to perform the task

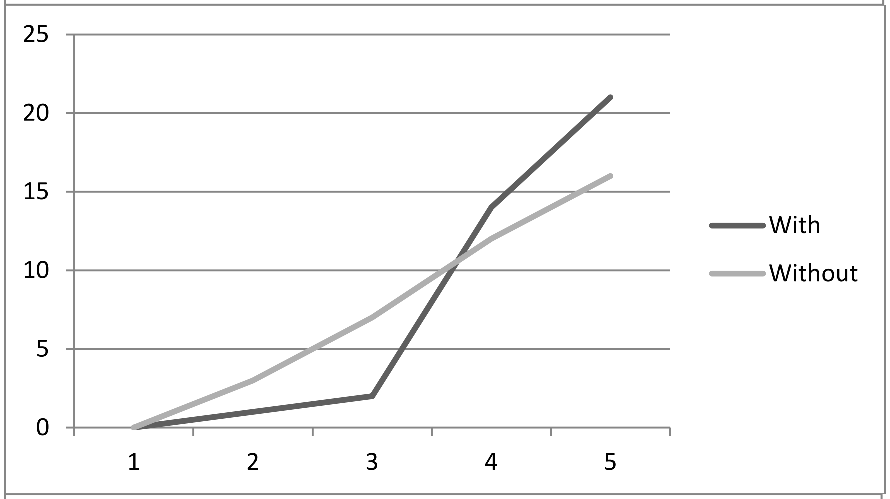


16.3 Appendix Three – Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Three

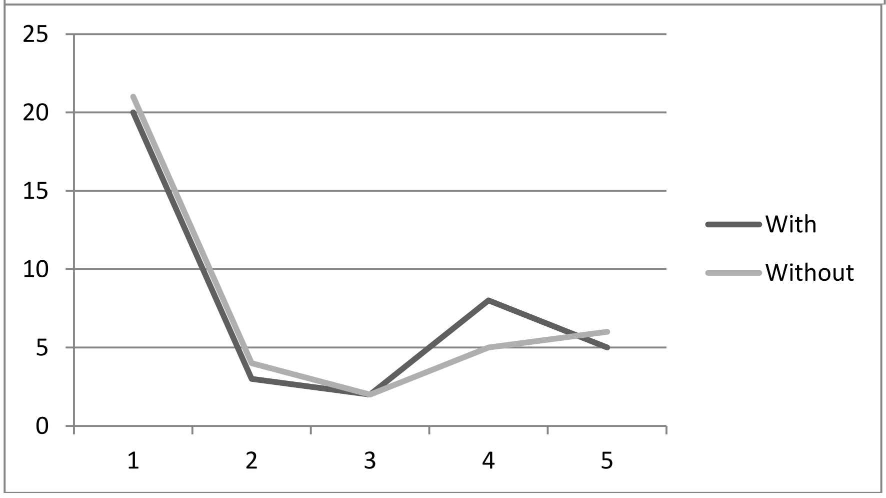
I feel that the task was easy to perform



I feel that the mouse was suitably responsive for the task

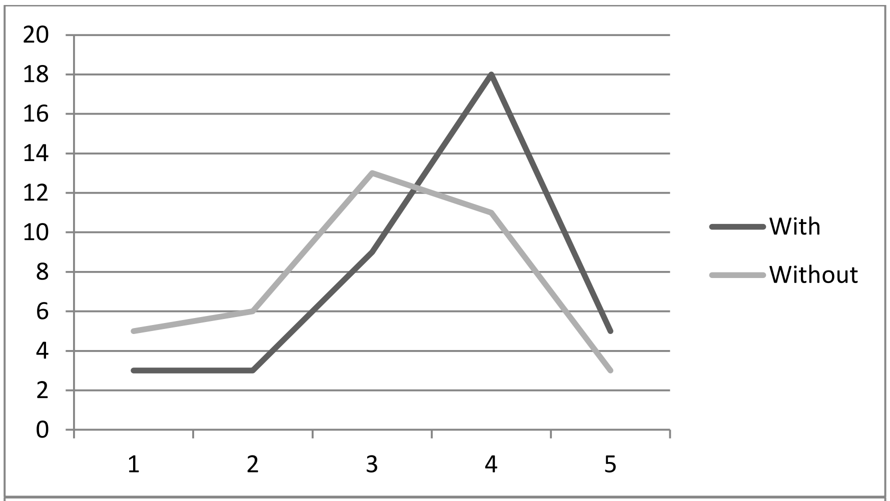


I know how to change the computer settings so that the mouse was more suited to perform the task

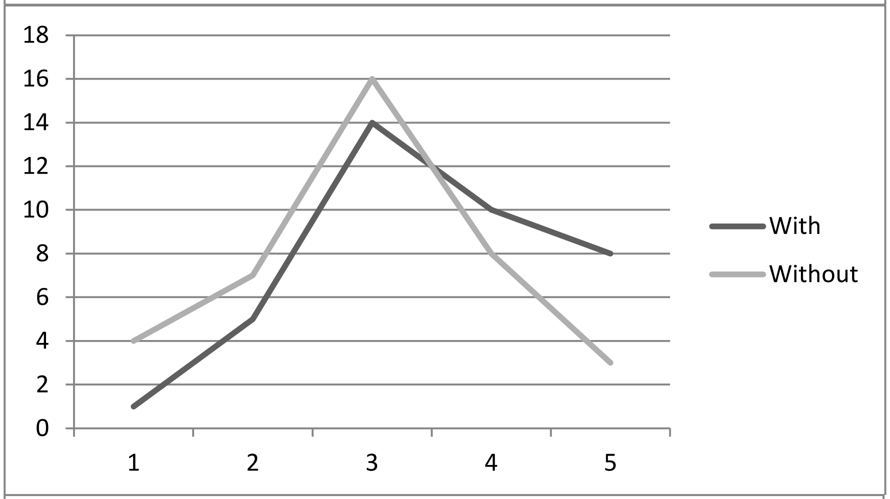


16.4 Appendix Four – Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Four

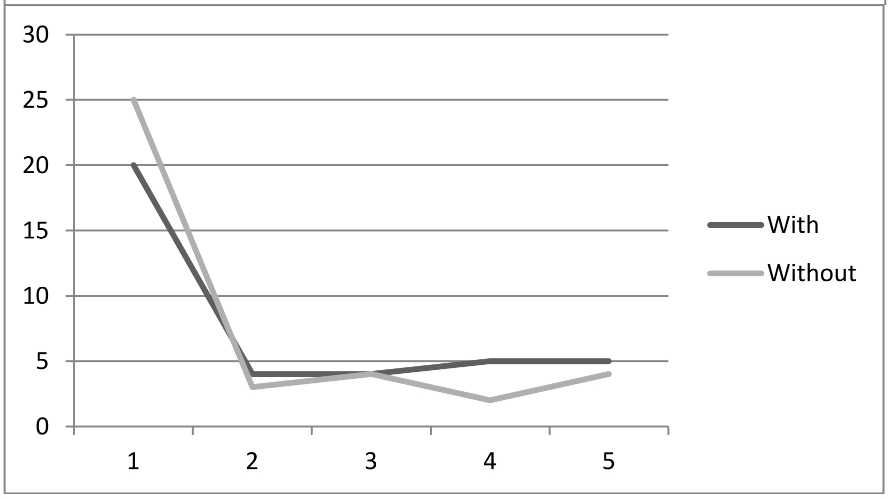
I feel that the task was easy to perform



I feel that the mouse was suitably responsive for the task

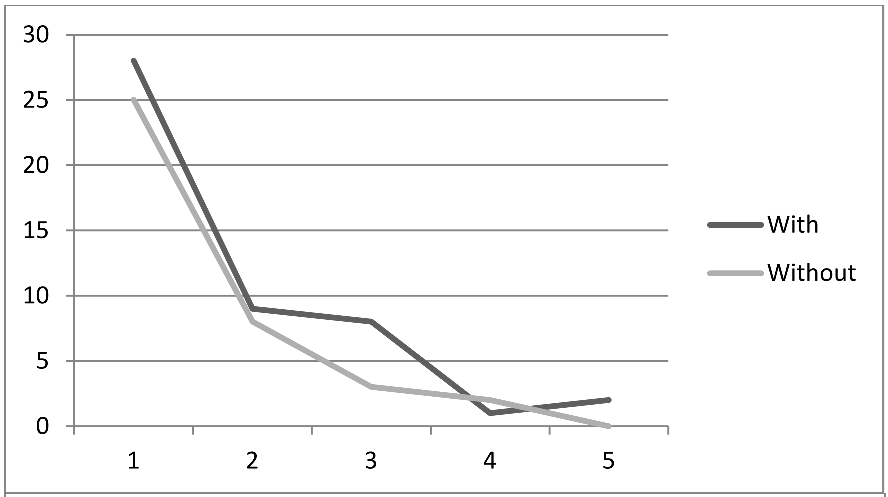


I know how to change the computer settings so that the mouse was more suited to perform the task

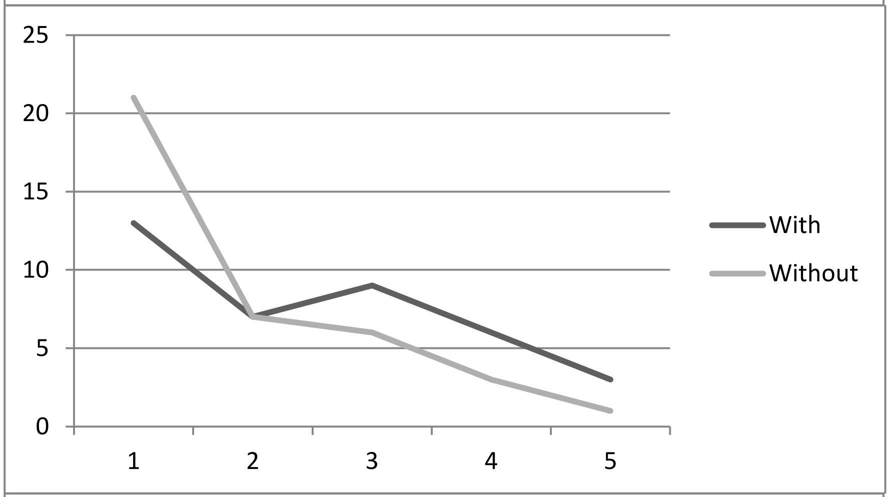


16.5 Appendix Five – Graphical Breakdown of Subjective Results for Multiple Plug-in Study, Task Five

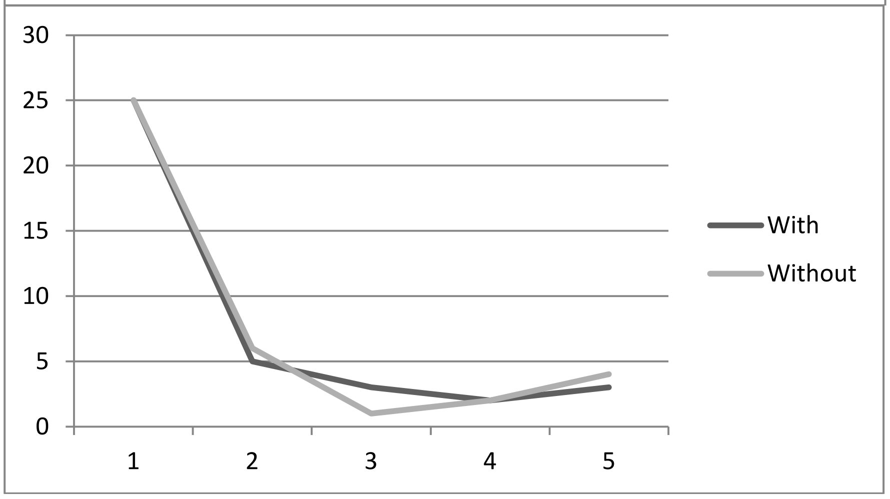
I feel that the task was easy to perform



I feel that the mouse was suitably responsive for the task



I know how to change the computer settings so that the mouse was more suited to perform the task



16.6 Tasks for Participants in Electronic Notes Study

16.6.1 Introduction

Thank you for taking part in this experimental study. This document outlines the tasks to be completed.

For these tasks, you do not need to know in advance how to do them – you will have either electronic or paper documentation provided for you giving the instructions. A star by an instruction indicates that the procedure has been given. This study is to determine whether electronic or paper documentation is better for this particular set of tasks – ***it is the documentation under examination, not you****.*

Please feel free to add your own notes to the documentation you have been provided if you feel it would be helpful.

16.6.2 Part One – Microsoft Word

1. Open up Sample Document.doc which is located on the desktop
2. Format the document, according to the following rules:
  - **a.** Find each occurrence of the word 'Kumquat' in the document, and make it bold.
  - **b.** Change the style of the words Abstract, Introduction, 'Table of Data', Discussion, and Conclusion to 'Heading 1'
  - **c.** Increase the indentation of the body text a level

3. Add a header to the document with the text 'Electronic Notes'
4. Add a footer to the document containing the text 'University of Dundee'
5. Add some graphics
  - **a.** Add the logo.gif file from the desktop
  - b. Move it to the top right hand side of the document.

6. Save the document you have created as a Word XML document.

On completion of each of these tasks (1-6), the researcher will make a note of when the task was completed and how often you referred to the documentation. Please remember that it is the relative quality of documentation we are assessing, not you – take whatever time you need.

16.6.3 Part Two – Microsoft Word and Internet Explorer

For this part of the study, you are creating a document for someone else to use, containing a list of details you have found.

1. Create a new Microsoft Word document, and give it whatever name you like.
2. In Internet Explorer, navigate to the Computing webpage of Dundee University.
    a. Find the telephone number and email address of Michael Heron
    b. Copy and paste this information into your word document.
3. Navigate to the BBC Scotland webpage in Internet Explorer.
    a. Get the front page of Scottish news
    b. Click on the link for the main story
    c. Copy the headline of the story
    d. Swap back to Microsoft Word
    e. Paste the text into your document.
    f. Apply the style 'heading 2' to the text you just pasted.
    g. Swap back to internet explorer
    h. Copy the text of the story
    i. Swap back to Microsoft Word
    j. Paste the text of the story underneath the heading you created for it.
    k. Swap back to Internet Explorer
    l. Copy the image that goes with the story
    m. Swap back to Microsoft Word
    n. Paste that image into your document.
    o. Set the layout of the image to be 'square'
    p. Move the image to the top right hand side of the document.
4. Navigate to the Amazon website in Internet Explorer.
    a. For each of the books listed in your provided documentation, find the ISBN number and paste it into your word document.
5. Navigate to Wikipedia in Internet Explorer.
    a. Find the page for the first book you were asked to find.
    b. Copy the introductory text from wikipedia
    c. Swap back to Microsoft Word
    d. Paste the text into your document underneath the ISBN for the book.
6. Save this document you have created as an HTML page.

On completion of each of these tasks (1-6), the researcher will make a note of when the task was completed and how often you referred to the documentation. Please remember that it is the relative quality of documentation we are assessing, not you – take whatever time you need.

16.7 Instructions for Participants in Electronic Notes Study

Finding and Bolding Each Occurrence of a Word

1. At the very right hand edge of the toolbar at the top is a button marked ‘find’ (it’s in a section called editing) – it has a pair of binoculars as an icon.
2. Press this button, and it brings up a new window asking you to enter the word you wish to find.
3. Type the word you wish to find into that box and press ‘find next’
4. Once it has highlighted the word, press ‘cancel’ to return to the document.
5. While the word is highlighted, press the button for making the text bold. It’s in the second section of the toolbar (marked Font), and looks like the letter B.
6. Go back to step one. Repeat steps one to five until all incidences of the have been made bold.

Changing The Style Of Words

1. At the very right hand edge of the toolbar at the top is a button marked ‘find’ (it’s in a section called editing) – it has a pair of binoculars as an icon.
2. Press this button, and it brings up a new window asking you to enter the word you wish to find.
3. Type the word you wish to find into that box and press ‘find next’
4. Once it has highlighted the word, press ‘cancel’ to return to the document.
5. While the word is highlighted, press the button for setting the style to be Heading 2. It’s in the fourth section of the toolbar, marked ‘styles’
6. Go back to step one. Repeat for each word you wish to change the style of.

Adding a Header

1. At the very top of the screen is a series of words such as ‘Home’ , ‘Insert’ and ‘Page Layout’
2. Click on the one that says ‘insert’. It’ll change the toolbar underneath.
3. Click on the button marked ‘Header’. It’s in the section titled ‘Header & Footer’, fifth from the left of the screen.
4. This will open up another window – click on the option marked ‘blank’.
5. This will cause the window to jump to the top of the document.
6. Type the text to appear along the top of the Document.
7. Press the button at the far right of the screen marked ‘close header and footer’

Adding a Footer

1. At the very top of the screen is a series of words such as ‘Home’ , ‘Insert’ and ‘Page Layout’
2. Click on the one that says ‘insert’. It’ll change the toolbar underneath.
3. Click on the button marked Footer. It’s in the section titled ‘Header & Footer’, fifth from the left of the screen.
4. This will open up another window – click on the option marked ‘blank’
5. This will cause the window to jump to the bottom of the document.
6. Type the text you wish to appear along the bottom of the document.
7. Press the button at the far right of the screen marked ‘close header and footer’

Adding Graphics

1. At the very top of the screen is a series of words such as 'Home' , 'Insert' and 'Page Layout'
2. Click on the one that says 'insert'. It'll change the toolbar underneath.
3. Press the button marked 'picture'. It's in the section marked 'Illustrations', third from the left.
4. This will open up the insert picture window. Navigate to the directory that contains the file you wish to add and click 'insert'
5. This will change the toolbar along the top.
6. Press the button marked 'position'
7. Choose the option that moves the image to the top right corner of the document.

Saving A File As A Different Type

1. Click on the large Microsoft office logo at the top left of the screen. It's in a circle and is coloured red, yellow, blue and green.
2. Choose the 'save as' option. This will open up the save window.
3. Click on the 'save as type' option and choose the type in which you wish to save.
4. Press the save button.

Creating A New Document

1. Click on the large Microsoft office logo at the top left of the screen. It's in a circle and is coloured red, yellow, blue and green.
2. Press the 'new' button.
3. Select 'blank document'
4. Press 'create'

Navigating to a Web Page

1. Open up Internet Explorer
2. Type the web address into the address bar (that's the one just to the right of the back and forwards arrows at the top)

To Copy Text from Microsoft Word

1. Select the text you wish to make a copy of.
2. Press the button marked 'copy' in the section of the toolbar marked 'Clipboard'
3. This will make a copy of the text and store it in the computer's memory.

To Copy Text from Internet Explorer

1. Select the text you wish to make a copy of.
2. Press the keys 'ctrl' and 'c' together.
3. This will make a copy of the text and store it in the computer's memory.

To Copy Pictures from Internet Explorer

1. While your mouse is over the picture you wish to copy, press the right mouse button.
2. Choose 'copy' from the menu that appears.
3. This will make a copy of the graphic and store it in the computer's memory.

To Paste to Microsoft Word

1. In Microsoft Word, press the button marked ‘paste’ in the section of the toolbar marked ‘Clipboard’

To Paste to Internet Explorer

1. In Internet Explorer, press the keys ‘ctrl’ and ‘z’ together.

16.8 Code for Double Click Plug-in

```
package accessframework;


public class DoubleClickPlugin extends AccessPlugin {

    private int current;
    private long lastClick;
    private int threshold;
    private int counter;
    private int offset = 100;
    private int motion = 0;

    public DoubleClickPlugin (AccessEngine e) {
      super (e);
      current = getEngine().getContext().getDoubleClickDelay();
      threshold = 2;
    }

    public String queryLogDetails() {
        return "Threshold (" + threshold   + "), Current (" + current  + ")";
    }

    public void positiveReinforcement() {
        threshold -= 1;
    }

    public void negativeReinforcement() {
        current -= offset;

        getEngine().getContext().setDoubleClickDelay (current);
        threshold += 1;
    }

    public void mouseMoved(long time, int x, int y) {
        // If the mouse moves too much, reset the last click - this wasn't
        // an attempt to double click at all.
        if (lastClick == 0) {
            return;
        }

        if (motion > 10) {
            System.out.println ("Reseting click");
            lastClick = 0;
            motion = 0;
        }
        motion += 1;
    }

    public void mouseClicked (long time, String button) {
        long currentClick;
        long delay = 0;
        currentClick = System.currentTimeMillis();

        delay = currentClick - lastClick;

        if (delay >= 1000) {
            lastClick = currentClick;
            return;
        }

        // The delay between these clicks was greater than the
        // double click threshold;
        if (delay > current) {
            counter += 1;
        }
        lastClick = 0;
    }

    public void keyTyped (long time, String key) {
    }

    public void create() {
    }

    public boolean wantsToMakeCorrection() {
      if (counter >= threshold && offset > 0) {
          return true;
      }
      return false;
    }

    public void performCorrection() {
        current += offset;

        getEngine().getContext().setDoubleClickDelay (current);
        counter = 0;
    }

    public String getName() {
      return "Double Click Plugin";
    }

    public void heartbeat() {
    }

    public void reset() {
    }

    public void invokePlugin() {
    }

    public Object getState() {
      return null;
    }

    public void setState(Object ob) {
    }

    public void tick() {
    }

    public void saveMe() {
    }

    public void loadMe() {
    }

    public String getMessage() {
        return "Double-clicking has been made a little easier to do.";
    }
}
```

16.9 Base for Pointer Size and Mouse Trails Plug-in

```
package accessframework;

import java.util.*;
import java.awt.*;

public class MouseLocatorBase extends AccessPluginMouse {

    ArrayList<PointTime> points;
    private boolean beenScolded;
    private long now, last;
    private int tremorThreshold;
    private int shakeThreshold;
    private int shakeCounter;
    private int cornerLength = 20;
    private int cornerHeight = 20;
    private int cornerCounter;
    Dimension screen;

    public MouseLocatorBase(AccessEngine e) {
        super(e);

        Toolkit toolkit = Toolkit.getDefaultToolkit();

        points = new ArrayList<PointTime>();

        shakeCounter = 0;
        shakeThreshold = 2;
        tremorThreshold = 80;

       screen = toolkit.getScreenSize();
    }

    public boolean getIsInRectangle (int x, int y, int len, int ht,
      int mx, int my) {
        if (mx < x || my < y) {
            return false;
        }

        if (mx > x + len || my > y + ht) {
            return false;
        }

        return true;
    }

    public boolean getIsInCorner (int x, int y) {
        if (getIsInRectangle (0, 0, cornerLength, cornerHeight, x, y)) {
            return true;
        }

        return false;
    }

    public void registerPoint (int x, int y) {
        now = System.currentTimeMillis();
        points.add(new PointTime(x, y, now));
    }

    public void positiveReinforcement() {
        tremorThreshold += 1;
        shakeThreshold -= 1;
    }

    public void negativeReinforcement() {
        tremorThreshold -= 1;
        shakeThreshold += 1;
    }

    public String queryLogDetails() {
        return "Tremor (" + tremorThreshold + "), Shake (" + shakeThreshold + ")";
    }

    public int getDistance(Direction d1, Direction d2) {
        int counter = 0;
        int current = 0;
        boolean inc = true;
        boolean dir;

        current = d1.getDir();

        if (d1 == Direction.UNKNOWN || d2 == Direction.UNKNOWN) {
            return 0;
        }

        while (current != d2.getDir()) {
            current += 1;

            if (inc) {
                counter += 1;
            } else {
                counter -= 1;
            }

            if (counter >= 4) {
                inc = false;
            }

            if (current > Direction.NORTHWEST.getDir()) {
                current = Direction.NORTH.getDir();
            }
        }

        if (inc) {
            return counter;
        }
        else {
            return counter * -1;
        }
    }

    public Direction getDirection(PointTime p1, PointTime p2) {
        int xDir, yDir;

        xDir = p1.getX() - p2.getX();
        yDir = p1.getY() - p2.getY();

        if (xDir < 0 && yDir == 0) {
            return Direction.WEST;
        }

        if (xDir > 0 && yDir == 0) {
            return Direction.EAST;
        }

        if (xDir == 0 && yDir < 0) {
            return Direction.NORTH;
        }

        if (xDir == 0 && yDir > 0) {
            return Direction.SOUTH;
        }

        if (xDir < 0 && yDir < 0) {
            return Direction.NORTHWEST;
        }

        if (xDir < 0 && yDir > 0) {
            return Direction.SOUTHWEST;
        }

        if (xDir > 0 && yDir < 0) {
            return Direction.NORTHEAST;
        }

        if (xDir > 0 && yDir > 0) {
            return Direction.SOUTHEAST;
        }

        return Direction.UNKNOWN;
    }

    public int getAbruptness(Direction a, Direction b) {
        Direction high, low;
        int val;

        if (a.getDir() > b.getDir()) {
            high = a;
            low = b;
        } else if (b.getDir() > a.getDir()) {
            high = b;
            low = a;
        } else {
            high = a;
            low = b;
        }

        val = Math.abs (getDistance(high, low));

        if (val == 4) {
            return 2;
        }
        if (val > 2) {
            return 1;
        }

        return 0;

    }

    public ArrayList<Direction> flattenDirections (ArrayList<Direction> dirs) {
        ArrayList<Direction> flattened = new ArrayList<Direction>();
        Direction last, current;

        for (int i = 1; i < dirs.size(); i++) {
            last = dirs.get(i-1);
            current = dirs.get(i);

            if (current != last) {
                flattened.add (current);
            }
        }

        return flattened;
    }

    public boolean detectSpiral (ArrayList<PointTime> points) {
        ArrayList<Direction> normalisedDirs = new ArrayList<Direction>();
        Direction nowD;
        PointTime tmp1, tmp2;
        int count, val, dist;
        boolean dir, lastDir = false;
        if (points.size() < 10) {
           return false;
        }

        try {
            for (int i = 0; i < points.size() - 1; i++) {
                tmp1 = points.get(i);
                tmp2 = points.get(i + 1);

                nowD = getDirection(tmp1, tmp2);

                normalisedDirs.add (nowD);
            }

            normalisedDirs = flattenDirections (normalisedDirs);

            count = 0;

            for (int i = 1; i < normalisedDirs.size(); i++) {
                if (normalisedDirs.get(i) == normalisedDirs.get(i-1)) {
                    continue;
                }

                val = getDistance (normalisedDirs.get(i), normalisedDirs.get(i-1));
                dist = Math.abs (val);

                if (val < 0) {
                    dir = false;
                }
                else if (val > 0) {
                    dir = true;
                }
                else {
                    continue;
                }


                if (dist > 2) {
                    continue;
                }

                if (dir == lastDir) {
                    count += 1;
                }
                else {
                       count = 0;
                 }

                lastDir = dir;

                if (count >= 4) {
                    return true;
                }
            }
        }
        catch (Exception ex) {
            System.out.println (ex.getMessage());
        }
        return false;
    }

    public boolean detectDirectionChange(ArrayList<PointTime> points) {
        PointTime tmp1, tmp2;
        Direction lastd = Direction.UNKNOWN, nowd;
        int numDirChanges = 0;
        int abruptnessCounter = 0;
        double percent;
        double avgAbrupt, overall;

        try {
            // Don't bother if we have less than 50 data points
            if (points.size() < 50) {
                return false;
            }

            // Discard the tails of the movement.
            for (int i = 0; i < points.size() - 2; i++) {
                tmp1 = points.get(i);
                tmp2 = points.get(i + 1);

                nowd = getDirection(tmp1, tmp2);

                if (lastd != Direction.UNKNOWN && nowd != lastd) {
                    if (Math.abs (getDistance(nowd, lastd)) > 1) {
                        numDirChanges += 1;
                        abruptnessCounter += getAbruptness(nowd, lastd);
                    }
                }

                lastd = nowd;
            }

            // Get the percentage of movements that were changes.

            percent = ((double) numDirChanges / (double) points.size()) * 100.0;
            avgAbrupt = ((double) abruptnessCounter /
                (double) points.size()) * 100.0;
            overall = avgAbrupt * percent;


            if (overall > 5) {
                return true;
            }
        }
        catch (Exception ex) {
            System.out.println (ex.getMessage());
        }
        return false;
    }

    public int getCursorSize() {
        return getEngine().getContext().getCursorSize();
    }

    public void keyTyped(long time, String key) {
    }

    public void create() {
    }

    public void clearPoints() {
        points.clear();
    }

    public ArrayList<PointTime> getPoints() {
        return points;
    }

    public void mouseMoved(long time, int x, int y) {
        now = System.currentTimeMillis();

        if (now - last > 100) {
            if (getIsInCorner (x, y)) {
                shakeCounter += 1;
            }
            else if (detectDirectionChange(points) == true) {
                shakeCounter += 1;
            }
            else if (detectSpiral(points) == true) {
                shakeCounter += 1;
            }

            clearPoints();
        }

        registerPoint (x, y);
        last = now;
    }

    public void mouseClicked(long time, String button) {
        clearPoints();
    }

    public boolean wantsToMakeCorrection() {
        if (shakeCounter >= shakeThreshold) {
            return true;
        }
        return false;
    }

    public void performCorrection() {
            shakeCounter = 0;
            clearPoints();
    }

    public String getName() {
        return "Mouse Locator Base";
    }

    public void heartbeat() {
    }

    public void invokePlugin() {
    }

    public Object getState() {
        return null;
    }

    public void setState(Object ob) {
    }

    public void tick() {
    }

    public void saveMe() {
    }

    public void loadMe() {
    }

}
```

16.10 Mouse Trails Plug-in

```java
package accessframework;

import java.awt.geom.Point2D;
import java.util.*;

public class MouseTrailsPlugin extends MouseLocatorBase {
    private int previousSize, lastCorrection;
    private boolean switchedOn;

    public MouseTrailsPlugin (AccessEngine e) {
        super(e);
        previousSize = 0;
    }

    public void negativeReinforcement() {
        super.negativeReinforcement();

        if (switchedOn) {
            getEngine().getContext().switchOffMouseTrails();
        }
        else {
            previousSize -= 1;
            getEngine().getContext().switchOnMouseTrails(previousSize);
        }
    }

    public boolean wantsToMakeCorrection() {
        if (previousSize > 7) {
            return false;
        }

        return super.wantsToMakeCorrection();
    }

    public void performCorrection() {
        if (getEngine().getContext().queryMouseTrails() == false) {
            previousSize = 3;
            getEngine().getContext().switchOnMouseTrails (previousSize);
            switchedOn = true;
        }
        else {
            previousSize += 1;
            getEngine().getContext().switchOnMouseTrails (previousSize);
            switchedOn = false;

        }
        super.performCorrection();
    }

    public String getName() {
        return "Mouse Trails Plugin";
    }

    public String getMessage() {
        if (switchedOn) {
            return "You should find that your cursor leaves a " +
              "trail when you move it now.";
        }
        else {
            return "The length of your mouse trails has been increased.";
        }
    }
}
```