

Understanding and Improving Navigation Within Electronic Documents

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Doctor of Philosophy
in the
University of Canterbury
by
Jason Alexander

Supervision and Examination Committee

Professor Andy Cockburn	Supervisor and Internal Examiner
Dr. Kenton O'Hara	External Examiner
Dr. George Buchanan	External Examiner
Associate Professor Tim Bell	Co-supervisor

Department of Computer Science and Software Engineering
University of Canterbury
2009

To my family and friends.

Abstract

ELECTRONIC documents form an integral part of the modern computer age—virtually all personal computers have the ability to create, store and display their content. A connection to the Internet provides users with an almost endless source of documents, be they web-pages, word-processor files or emails.

However, the entire contents of an electronic document are often too large to be usefully presented on a user's screen, at a single point in time. This issue is usually overcome by placing the content inside a scrolling environment. The view onto the document is then modified by directly adjusting a scrollbar or by employing tools such as the mousewheel or paging keys. Applications may also provide methods for adjusting the document's zoom and page layout.

The scrollbar has seen widespread adoption, becoming the default tool used to visualise large information spaces. Despite its extensive deployment, researchers have little knowledge on how this and related navigation tools are used in an everyday work environment. A characterisation of users' actions would allow designers to identify common behaviours and areas of inefficiency as they strive to improve navigation techniques.

To fill this knowledge gap, this thesis aims to understand and improve navigation within desktop-based electronic documents. This goal is achieved using a five step process. First, the literature is used to explore document navigation tasks and the tools currently available to support electronic document navigation. Second, a software tool called AppMonitor, that logs users' navigation actions, was developed. Third, AppMonitor was deployed in a longitudinal study to characterise document navigation actions in Microsoft Word and Adobe Reader. Fourth, to compliment this study, two task-centric observations of electronic document navigation were performed, to probe the reasons for navigation tool selection. Finally, the Footprints Scrollbar was developed to improve one common aspect of navigation—within-document revisitation.

To begin, two areas of current knowledge in this domain are overviewed: paper and electronic document navigation and electronic document navigation

tools. The literature review produced five categories of document navigation tasks: ‘overviewing and browsing’, ‘reading’, ‘annotating and writing’, ‘searching’ and ‘revisitation’. In a similar fashion, electronic document navigation tools were reviewed and divided into eight categories: core navigation tools (those commonly found in today’s navigation systems), input devices, scrollbar augmentations, content-aware navigation aids, visualisations that provide multiple document views, indirect manipulation techniques, zooming tools and revisitation tools.

The literature lacked evidence of an understanding of how these current document navigation tools are used. To aid the gathering of empirical data on tool use, the AppMonitor tool was developed. It records user actions in unmodified Windows applications—specifically for this research, Microsoft Word and Adobe Reader. It logs low-level interactions such as “left mouse button pressed” and “Ctrl-f pressed” as well as high level ‘logical’ actions such as menu selections and scrollbar manipulations. It requires no user input to perform these tasks, allowing study participants to continue with their everyday work.

To collect data to form a characterisation of document navigation actions, 14 participants installed AppMonitor on their computer for 120 days. This study found that users primarily employ the mousewheel, scrollbar thumb and paging keys for navigation. Further, many advanced navigation tools that are lauded for their efficiency, including bookmarks and search tools, are rarely used.

The longitudinal study provided valuable insights into the use of navigation tools. To understand the reasons behind this tool use, two task-centric observations of electronic document navigation were conducted. The first asked participants to perform a series of specific navigation tasks while AppMonitor logged their actions. The second was performed as a series of interactive sessions, where users performed a particular task and were then probed on their tool choice. These two studies found that many users are not aware of the advanced navigation tools that could significantly improve their navigation efficiency.

Finally, the characterisations highlighted within-document revisitation as a commonly performed task, with current tools that support this action rarely used. To address this problem, the analysis, design and evaluation of a Footprints Scrollbar is presented. It places marks inside the scrollbar trough and provides shortcuts to aid users return to previously visited locations. The Footprints Scrollbar was

significantly faster and subjectively preferred over a standard scrollbar for revisitation tasks.

To summarise, this thesis contributes a literature review of document navigation and electronic document navigation tools; the design and implementation of AppMonitor—a tool to monitor user actions in unmodified Windows applications; a longitudinal study describing the navigation actions users perform; two task-centric studies examining why actions are performed; and the Footprints Scrollbar, a tool to aid within-document revisitation tasks.

Publications Arising from this Thesis

PORTIONS of the work contained in this thesis have appeared as peer-reviewed publications in conferences and journals. The following list details these publications. The chapter from which the material was used is noted in brackets.

1. **Alexander, J.** and Cockburn, A. Characterising Electronic Document Use, Reuse, Coverage and Multi-Document Interaction. In *NZCSRSC '08: New Zealand Computer Science Research Student Conference 2008*, (Christchurch, New Zealand, 2008), 1–8. (Chapter 5).
2. **Alexander, J.**, Cockburn, A., and Lobb, R. AppMonitor: A Tool for Recording User Actions in Unmodified Windows Applications. *Behavior Research Methods*, 40(2):413–421, May 2008. (Chapter 4).
3. **Alexander, J.** and Cockburn, A. An Empirical Characterisation of Electronic Document Navigation. In *GI '08: Proceedings of Graphics Interface 2008*, (Windsor, Ontario, Canada, 2008), Canadian Information Processing Society, 123–130. **A.J. Sweeny Award for Best Student Paper at GI 2008**. (Chapter 5).
4. **Alexander, J.**, Cockburn, A., Fitchett S., Gutwin, C. and Greenberg S. Revisiting Read Wear: Analysis, Design and Evaluation of a Footprints Scrollbar. In *CHI '09: SIGCHI Conference on Human Factors in Computing Systems, 2009*, (Boston, MA, USA, 2009), ACM, 1665–1674. (Chapter 7).
5. **Alexander, J.** and Cockburn, A. Deploying an Application Logger to Characterise Document Navigation. In *CHI '09 Workshop: Best Practices in Longitudinal Research*. (Chapter 4).

During my doctoral studies I also participated in a research project conducted at IBM Almaden, resulting in a further publication:

6. Cockburn, A., Kristensson, P., **Alexander, J.** and Zhai, S. Hard Lessons: Effort-Inducing Interfaces Benefit Spatial Learning. In *CHI '07: SIGCHI Conference on Human Factors in Computing Systems, 2007*, (San Jose, California, USA, 2007), ACM, 1571–1580.

Declaration and Technical Acknowledgements

EXCEPT as noted below, the work contained within this thesis is all my own. My supervisor, Professor Andy Cockburn, provided the inspiration for this research. His ideas and contributions are intricately weaved into the entire thesis.

Chapter 4—AppMonitor: Dr. Richard Lobb was responsible for establishing the groundwork for AppMonitor. He validated the feasibility of the concept and produced a basic working prototype for Microsoft Word. Small portions of an unpublished technical document describing the programming concepts used in AppMonitor appear in publication #2.

Chapter 7—The Footprints Scrollbar: This work was a collaboration between five authors. Proof-of-concept work was conducted by Stephen Fitchett during a summer research project. I designed and implemented experiments one and two; participants were run through these experiments by a summer research student in Professor Carl Gutwin’s lab at the University of Saskatchewan. Experiment four was run by Andy Cockburn. Author order on publication #4 is indicative of the ideas, discussions, feedback and contributions to all aspects of the paper.

Publications #1, #3 and #5: Were all written in collaboration with Andy Cockburn. These papers were primarily my work, however, they were based on research that had significant input from Andy. He provided ideas, proof-read and made suggestions for improvements in all of these papers.

My gratitude is extended to everyone who was involved in research collaborations that contributed to this thesis.

Ethical Considerations

THE experiments and studies documented in this thesis involve human participants. Their privacy, comfort and well-being was carefully considered during this research. All participants maintained the right to withdraw and have their data destroyed at any point.

The work undertaken in this thesis was approved by the University of Canterbury's Human Ethics Committee (number HEC2008/130). A further information sheet outlining the longitudinal study documented in Chapter 5 was also provided to the ethics committee (see Appendix A).

Experiments one and two of Chapter 7 (evaluating the Footprints Scrollbar) were conducted at *The Interaction Lab*, at the University of Saskatchewan, and were covered by Professor Carl Gutwin's ethics approval.

All study participants signed consent forms before any experimental data, demographic information or survey responses were collected. Copies of these forms are reproduced in the appendices.

Acknowledgements

I wish to extend my thanks to the following people who have supported me during the course of my Ph.D. Without them, this thesis would not exist.

First, thank-you to my supervisor, Professor Andy Cockburn, for providing the original inspiration for this research, and for your patience, guidance and proof-reading of my work. Your passion for HCI, dedication to ‘good science’, enthusiasm to challenge the system and unrelenting attention to detail are an inspiration to me and to all of your students. Thank-you also to my co-supervisor, Associate Professor Tim Bell, for your guidance and sanity-checking throughout.

Thank-you to all of my family and friends for your unwavering support and encouragement. A special thank-you to my parents, Geraldine and Ian Alexander for your emotional (and financial) support and encouragement throughout this Ph.D. and my entire education. Thank-you also to my grandparents Doreen and Gerald Wright who instilled into me the importance of a good education from a very young age.

Thank-you to all of my friends who have provided sanity-saving distractions from my thesis and for understanding my sometimes over-exuberant commitment to this research. This is especially true of Jenny, Kay and Andrew for the countless dinners and movies; Callum, Jonelle and their daughter Jessalyn for teaching me that no-one is too old or too manly to play with a Barbie; the NZCSRSC ’08 team (Amali, Amanda, Andrew, Charles, Delio, Devon, Isaac, Jay, Moffat, Mohammad, Ray, Sascha, Sharon, Tim and Xianglin) who showed me the joy and comradeship of organising a conference; and Yalini for providing me with endless opportunities to hone my teaching skills. Thank-you to all of my other friends that have provided support (even if it was only your awe that I was actually doing this!).

To my past and present colleagues in the HCI research group, both those based in the HCI lab (Michael, Taher, Isaac, Susanne, David, Stephen, Philip) and those based at the HitLabNZ (Julian, Jörg, Oakley, Christina, Christiano): thank-you for the discussions, debates and constructive criticism during my research. Thanks

also to Michael for frequently offering yourself up as a model for what a Ph.D. student should *not* be. I can proudly hold my head high and say “I was not like Michael JasonSmith”.

To all of my colleagues, both staff and student, in the Department of Computer Science and Software Engineering at the University of Canterbury: thank-you for the stimulating conversations, intellectual (and sometimes not so intellectual) discussions, debates and hilarity during morning-tea and lunch breaks. You have made my time in the department that much more enjoyable. You should be proud of the friendly and supportive atmosphere in this department. I feel honoured to have been part of it.

I would also like to acknowledge the department’s technical staff for keeping the disks spinning and for not blacklisting me from the TAS system after my frequent job requests. Thanks also to Pete for demonstrating to me the number of problems that can be solved with the ‘`rm -rf *`’ command.

Thank-you to Professor Carl Gutwin at the University of Saskatchewan and Professor Saul Greenberg at the University of Calgary who both hosted me in their labs during June 2008. It was fantastic to be able to immerse myself in your enthusiasm and experience.

Finally, thank-you to all of the participants who volunteered to give up their time to take part in the studies and experiments that are documented in this thesis. Without you I would have no results.

This thesis was financially supported by the prestigious *Canterbury Scholarship*, awarded to the top eight applicants for doctoral scholarships at the University of Canterbury. Conference funding was partially supported by the *New Zealand Royal Society Marsden Grant* 07–UOC–013 and a *BuildIT Travel Fund for Ph.D. Students* award (one of only sixteen made each year).

Table of Contents

List of Tables	xxvii
List of Figures	xxix
Source Code Listings	xxxi
Chapter 1: Introduction	1
1.1 Problem Statement	2
1.2 Research Approach	3
1.3 Research Contributions	4
1.4 Structure of the Thesis	5
 I Document Navigation and Document Navigation Tools	 9
Chapter 2: An Overview of Paper and Electronic Document Navigation	11
2.1 Foundations	11
2.1.1 Document Navigation Sub-tasks	13
2.2 Navigation Tasks Within Paper and Electronic Documents	14
2.2.1 Overviewing and Browsing a Document	14
2.2.2 Reading	16
2.2.3 Annotating and Writing Documents	22
2.2.4 Searching Within A Document	26
2.2.5 Document Revisitation	29
2.3 Observing and Recording Document Navigation	30
2.3.1 Human-Centric Observation	30
2.3.2 Automated Logging Software	33
2.4 Summary and Discussion	37

Chapter 3:	A Review of Electronic Document Navigation Tools	39
3.1	Core Navigation Tools	41
3.1.1	The Scrollbar	41
3.1.2	The Mousewheel	42
3.1.3	Paging	43
3.1.4	Rate-based Scrolling (RBS)	44
3.1.5	Search Tools	44
3.1.6	Goto Tools	46
3.1.7	Zoom Tools	46
3.1.8	Panning	50
3.1.9	Document Layout	50
3.2	Input Devices	51
3.2.1	Mouse Activated Document Navigation	51
3.2.2	Mouse Appendages and Augmentations	53
3.2.3	Keyboard Based Navigation	54
3.2.4	Stylus-Based Input	56
3.2.5	SlideBar	56
3.3	Scrollbar Augmentations	57
3.3.1	Thumbnail-Enhanced Scrollbars (TES)	57
3.3.2	Scrollbar-Based Visualisations	59
3.3.3	Scrollbar Behaviour Modifications	63
3.4	Content-Aware Navigation Aids	64
3.4.1	Content-Aware Scrolling (CAS)	64
3.4.2	Intra-Document Links	64
3.4.3	Semantic Browsing	66
3.4.4	Content Visualisations	66
3.5	Document Visualisations that Provide Multiple Views	74
3.5.1	Split-Views	75
3.5.2	Overview+Detail	75
3.5.3	Focus+Context	78
3.6	Navigation by Indirect Manipulation	79
3.6.1	Rate-Controlled Techniques	79
3.6.2	Navigation Using Rotational Actions	81
3.6.3	Two-handed Navigation	83

3.6.4	Perspective View Navigation	84
3.6.5	Navigation Using Eye-Tracking	84
3.7	Zoom Tools	85
3.7.1	Semantic Zooming	85
3.7.2	Parallel Scale Manipulation	86
3.7.3	Automatic Zooming	87
3.8	Revisitation Tools	87
3.8.1	User-Defined Bookmarks	87
3.8.2	Electronic Dog-Ears	88
3.8.3	Click-Through Navigation	88
3.8.4	Previous View and Next View	89
3.9	Navigation Techniques on Mobile Devices	89
3.9.1	Small-Screen Devices	90
3.9.2	E-book Readers	91
3.10	Evaluating Document Navigation Tools and Techniques	92
3.10.1	What to Measure?	92
3.10.2	Evaluation Frameworks	94
3.10.3	Evaluations of Document Navigation Tools and Techniques	95
3.10.4	Modeling Tool Performance	96
3.11	Summary and Discussion	99
3.11.1	Adoption of Improvements into Document Navigation Systems	99
3.11.2	Issues Associated with Document Navigation Tool Evaluations	101

II Characterising Electronic Document Navigation 103

Chapter 4: AppMonitor: A Tool for Recording Document Navigation Actions in Unmodified Windows Applications 105

4.1	Motivation for Creating an Application Logger	106
4.2	System Requirements	106
4.3	System Architecture	108
4.3.1	Interaction with Monitored Applications	108
4.3.2	Event Scheduling	110

4.3.3	The Window Hierarchy	110
4.3.4	Low-level Event Logging	114
4.3.5	High-level Event Logging—WinEvents	117
4.3.6	Polling	117
4.3.7	User Interface	118
4.4	Portability and Extendability	119
4.5	Implementation Issues and Lessons Learned	120
4.5.1	Defensive Programming Techniques	120
4.5.2	Accessing a Microsoft Word Document's Page Length . .	121
4.5.3	Using the Active Accessibility Interfaces with Adobe Reader	122
4.5.4	Delaying Logging until an Application's Window Struc- ture is Stable	122
4.5.5	Unimplemented Functions	123
4.6	Configuration	124
4.7	Log Files	124
4.8	Parsing and Analysing Log Files	128
4.8.1	Dividing Log Files into Documents	128
4.8.2	Regular Expression Library	129
4.8.3	Decoding Log File Events	129
4.9	Summary	130

**Chapter 5: How do Users Navigate in Documents? An Empirical
Characterisation of Electronic Document Navigation 133**

5.1	Study Goals	134
5.2	Participants	134
5.3	Methodology	136
5.4	Data Analysis Parameters, Assumptions and Result Presentation .	136
5.4.1	Document Usage Sessions	136
5.4.2	Only Explicit Navigation Actions are Considered	137
5.4.3	Grouping Navigation Events into Navigation Actions . . .	138
5.4.4	Presentation of Results	139
5.5	Document Properties	141
5.5.1	Scale of the Study	141
5.5.2	Do users interact with the documents they open?	141

5.5.3	How often are documents re-opened?	142
5.5.4	How many pages are there in a document?	143
5.5.5	How long are documents open?	144
5.5.6	How long do users spend interacting with their documents?	145
5.6	Vertical Document Navigation	145
5.6.1	How should vertical document navigation be characterised?	146
5.6.2	A Categorisation of Navigation Actions	147
5.6.3	A Categorisation of Word and Reader's Tools	148
5.6.4	How are users' actions divided between the navigation categories?	148
5.6.5	Are users' actions consistent across applications?	150
5.6.6	How do individual tools contribute to each navigation category?	150
5.6.7	How far do navigation actions move?	150
5.6.8	What is the time duration of navigation actions?	153
5.6.9	What percent of users' time is spent navigating?	154
5.6.10	What is the velocity of navigation actions?	155
5.7	Analysis of Selected Vertical Document Navigation Tools	156
5.7.1	The Mousewheel	158
5.7.2	The Scrollbar Thumb	160
5.7.3	The Paging Keys	163
5.7.4	Panning Using Reader's Hand Tool	164
5.7.5	Bookmarks	165
5.7.6	Thumbnails	166
5.7.7	Use of Find and Goto Tools	166
5.7.8	Rate-based Scrolling	167
5.7.9	Classifying Users into Stereotypical Navigator Categories	168
5.8	Use of Other Navigation Tools	170
5.8.1	Horizontal Document Navigation	170
5.8.2	Page Layout Tools	170
5.8.3	Zoom Tools	173
5.9	Navigation Patterns	175
5.9.1	User Interest in Document Regions	175
5.9.2	Activity Patterns	177

5.9.3	Within-Document Revisitation	179
5.10	Further Observations	183
5.10.1	Use of Reviewing Tools	183
5.10.2	Document and Window Management Activities	184
5.10.3	Use of Menu Items, Toolbar Buttons and Keyboard Shortcuts	188
5.11	Discussion	196
5.11.1	Implications for Navigation Tool Designers	197
5.11.2	Implications for System Designers	199
5.11.3	Limitations	199
5.11.4	Future Areas for Similar Work	200
5.11.5	Recommendations for Researchers Conducting Similar Studies	200
5.12	Summary	201

Chapter 6: Why do Users Navigate in a Particular Manner? Two Task-Centric Observations of Electronic Document Navigation 203

6.1	Study Goals	204
6.2	Task Study	205
6.2.1	Apparatus	205
6.2.2	Experimental Design and Procedure	207
6.2.3	Participants	207
6.3	Interactive Sessions	209
6.3.1	Participants	209
6.3.2	Apparatus	209
6.4	Tasks	210
6.5	Results and Discussion	211
6.5.1	Interface Setup	211
6.5.2	Awareness of the Document's Length	213
6.5.3	Understanding a Document's Structure	215
6.5.4	Locating a Named Section	216
6.5.5	Locating a Page	217
6.5.6	Phrase Search	218

6.5.7	Finding an Answer	219
6.5.8	Revisitation	220
6.5.9	Region Comparison	223
6.5.10	Visual Search Techniques	225
6.5.11	Zoom	225
6.5.12	Extremity Shortcuts	227
6.5.13	Reading	229
6.5.14	Document Editing	229
6.5.15	Document Opening Methods	230
6.5.16	Familiarity with Advanced Navigation Tools	231
6.6	Task-Study Subjective Analysis	234
6.6.1	Task Difficulty	234
6.6.2	Freeform Comments	234
6.7	Comments from the Interviewees	235
6.7.1	Find and Search Tools	235
6.7.2	Code Editor Features	236
6.7.3	Zoom Tools	236
6.7.4	Tool Unpredictability	236
6.7.5	Navigation Style	237
6.8	Limitations	237
6.8.1	Generalisability	237
6.8.2	The Hawthorne Effect	238
6.8.3	Study Depth	238
6.9	Future Work	238
6.9.1	Users' Skill Levels	239
6.9.2	Task Selection	239
6.9.3	In-context Observations	239
6.10	Summary	240

III Improving Within-document Revisitation 243

Chapter 7: The Footprints Scrollbar 245

7.1	Summary of Current Revisitation Tools	246
7.2	Summary of Revisitation Log Analysis	248

7.3	Direction for Design	249
7.4	Effectiveness of Recency and Frequency Based-History Lists for Storing Revisitation Information	249
7.5	Experiment One: Can Recency Marks Help?	252
7.5.1	Participants and Apparatus	252
7.5.2	Experimental Design and Procedure	253
7.5.3	Results	254
7.5.4	Discussion	256
7.6	Experiment Two: Marking Overload	257
7.6.1	Experimental Setup	257
7.6.2	Results and Discussion	257
7.7	The Footprints Scrollbar Design	259
7.7.1	Marking Algorithms and Behaviour	259
7.8	Experiment Three: Footprints Scrollbar Evaluation	263
7.8.1	Experimental Method	263
7.8.2	Results	265
7.8.3	Use of Revisitation Tools in the Footprints Scrollbar	266
7.9	Experiment Four: Observational Study	267
7.9.1	Results	268
7.10	Discussion of Experiments Three and Four	269
7.10.1	Mark Colours	270
7.10.2	Digit Shortcuts	270
7.10.3	Back/Forward Keyboard Shortcuts	270
7.10.4	Lack of Control Over Mark Placement	270
7.11	Summary and Future Work	271

IV Discussion and Future Directions 273

Chapter 8: Discussion and Future Work 275

8.1	Progress on Research Objectives	275
8.2	Lessons for Practitioners	278
8.3	Research Generalisability	279
8.3.1	AppMonitor	279
8.3.2	Characterisations of Document Navigation	279

8.3.3	Revisitation and the Footprints Scrollbar	280
8.4	Document Navigation into the Future	280
8.5	Future Work	281
8.5.1	Understanding the Context of User Actions	282
8.5.2	Modelling User Actions	282
8.5.3	Development of Interaction Techniques	282
8.5.4	A Standardised Evaluation Framework	283
Chapter 9:	Conclusion	285
	References	287
	Appendices	319
Appendix A:	Information Sheet for Human Ethics Committee	321
Appendix B:	Applications Surveyed for Document Navigation Tool Support	325
Appendix C:	AppMonitor Log File Syntax	327
Appendix D:	Longitudinal Study Material	329
D.1	Information Sheet	329
D.2	Consent Form	331
D.3	Demographic Information Form	333
Appendix E:	Task-centric Observation Material	337
E.1	Consent Form	337
E.2	Instruction Sheet	339
E.3	Demographic Information Form	341
Appendix F:	Interactive Session Material	345
F.1	Consent Form	345
F.2	Interviewer Instruction and Question Script	347
F.3	Interactive Session Resources	360

Appendix G: Footprints Scrollbar Material	363
G.1 Marking Scrollbar Evaluation Consent Form	363
G.2 Electronic Demographic Information Form	365
G.3 Electronic Subjective Evaluation Form	365
G.4 Footprints Scrollbar Evaluation Consent Form	365
G.5 Footprints Scrollbar Evaluation Feedback Form	369
G.6 Footprints Scrollbar Observational Study Interviewer Instruction Sheet	372
G.7 Footprints Scrollbar Observational Study Scenario Questionnaire .	375

List of Tables

3.1	Application support of document navigation tools	40
3.2	Manual zoom tool classification	47
3.3	Methods for evaluating document navigation tool performance . .	93
3.4	Summary of document navigation tool evaluations	97
4.1	Keyboard modifier masks	115
4.2	Registry values to prevent ‘reading order’ dialog	123
5.1	Longitudinal study participants’ demographic information	135
5.2	Summary of document use and re-use	142
5.3	Categorisation of document navigation tools	148
5.4	Summary statistics for the use of the mousewheel	159
5.5	Summary statistics for use of the scrollbar thumb	161
5.6	Summary statistics for use of the paging keys	163
5.7	Summary statistics for use of the hand-tool in Adobe Reader . . .	165
5.8	Summary of find actions	167
5.9	Participants’ navigator categories	169
5.10	Summary of horizontal scrolling actions	171
5.11	Description and naming conventions of page layout tools	172
5.12	Changes made to the page layout during document interaction . .	173
5.13	Summary of zoom actions	174
5.14	Summary of the use of zoom tools	174
5.15	Summary of revisitation from the longitudinal study	181
5.16	Summary of the use of reviewing tools in Microsoft Word	184
5.17	Summary of document management activities	185
5.18	Summary of window management operations	187
5.19	Summary of menu hunting	189
5.20	The fifteen most frequently used shortcut keys in Microsoft Word .	196
5.21	The ten most frequently used shortcut keys in Adobe Reader . . .	197

6.1	Demographic information for participants in the task study	208
6.2	Tasks in the task-study and interactive sessions	210
6.3	Summary of interviewees' interface setups for Microsoft Word . .	212
6.4	Tools used to complete visual search tasks in Microsoft Word . . .	226
6.5	Tools used to complete visual search tasks in Adobe Reader	226
7.1	Summary of revisitation tool use from the longitudinal study . . .	248
7.2	NASA-TLX responses in experiment one	256
7.3	Example of the Footprints Scrollbar's back/forward behaviour . .	262
7.4	NASA-TLX responses in experiment three	267
7.5	Responses to subjective questions in experiment four	268
7.6	Rankings of the Footprints Scrollbar's revisitation tools	269
B.1	Applications surveyed for document navigation tool support . . .	326

List of Figures

2.1	Sub-tasks in a document navigation action.	13
3.1	Standard vertical scrollbar components	42
3.2	Rate-based scrolling icon	45
3.3	Microsoft Word's <i>Find and Replace</i> dialog box	45
3.4	Zoom controls	49
3.5	Two of Microsoft Word's document views	52
3.6	Common desktop input devices	53
3.7	Keyboard-based navigation	55
3.8	Adobe Reader's thumbnail-enhanced scrollbars	58
3.9	Scrollbar marks	62
3.10	Example bookmarking mechanisms	67
3.11	Semantic browsing controls	68
3.12	Context based search mechanisms	69
3.13	Diff tools	71
3.14	Microsoft Word's reviewing tools	73
3.15	Microsoft Word's split-view interface	76
3.16	Example overview+detail interfaces	77
3.17	RSVP rate control icon	81
4.1	AppMonitor's high-level architecture	109
4.2	AppMonitor's internal scheduling architecture	111
4.3	Spy++ displaying part of the window hierarchy	113
4.4	AppMonitor's event display window	118
4.5	AppMonitor's configuration dialog	126
5.1	Example box-and-whisker chart	140
5.2	Distribution of document re-openings	143
5.3	Mean document lengths	144
5.4	Mean document interaction times	146

5.5	Distribution of navigation actions into categories	149
5.6	Per-user categorisation of navigation actions, per application . . .	149
5.7	Distribution of navigation actions, by tool	151
5.8	Distribution of the mean distances moved by navigation actions . .	152
5.9	Mean percent of total distance navigated, by category	153
5.10	Mean distance navigated, by tool	154
5.11	Mean time period of navigation actions	155
5.12	Mean percent of participants' interaction time spent navigating . .	156
5.13	Mean navigation velocities, by tool	157
5.14	Summary of the attributes of navigation tool use	158
5.15	Period of mousewheel navigation actions	160
5.16	Period vs. distance of mousewheel navigation actions	161
5.17	Participants' mean distance navigated with the scrollbar thumb . .	162
5.18	Periods vs. distances of paging key actions	164
5.19	Distances of hand-tool navigation actions	166
5.20	Percent of documents in which particular regions were viewed . .	176
5.21	Mean percent of time spent in document regions	177
5.22	Generalised and example document positions over time	178
5.23	Distribution of navigation patterns over the documents' lifetimes .	180
5.24	The number of times positions are revisited within a document . .	182
5.25	Distribution of use of reviewing tools	184
5.26	Microsoft Office Word 2007's Ribbon	189
5.27	Participants' mean distribution of the use of Microsoft Word's menus	190
5.28	Participants' mean distribution of the use of Adobe Reader's menus	191
5.29	Context menu selections for Microsoft Word	193
5.30	Distribution of toolbar button use	194
6.1	Experimental interface for the task-based study	206
6.2	Document length task result summary	214
6.3	Revisitation task completion times	221
6.4	Tools used for revisitation tasks	222
6.5	Tools and task completion times for comparison tasks	224
6.6	Competency and frequency of use of Word's navigation tools . . .	232
6.7	Competency and frequency of use of Reader's navigation tools . .	233

7.1	Usefulness of different types and lengths of history list	251
7.2	Experiment one interface	253
7.3	Mean target acquisition times in experiment one	255
7.4	Mean revisit times in experiment two	258
7.5	The Footprints Scrollbar	260
7.6	Scrollbar thumb-filling animation	262
7.7	Experimental interface for the Footprints Scrollbar evaluation . . .	264
7.8	Mean revisit times in experiment three	265
G.1	Electronic demographic information form	366
G.2	Electronic subjective evaluation form	367

Source Code Listings

4.1	Top level window iteration to identify applications of interest . . .	112
4.2	Initialising a keyboard hook for Microsoft Word	115
4.3	Keyboard callback function for Microsoft Word	116
4.4	Function that determines which keycodes to ignore (not log) . . .	116
4.5	Defensive programming using a secured <code>printf</code> function	121
4.6	Sample AppMonitor log file, for a Microsoft Word session	125
4.7	Example regular expressions	129
4.8	Example script to extract <code>ScrollbarChanged</code> events	129
4.9	Function to convert <code>ScrollbarsChanged</code> events to a percent . . .	130
C.1	BNF syntax for AppMonitor log files	328

Chapter 1

Introduction

ELECTRONIC documents, be they word-processor outputs, instruction manuals or web-pages, form an integral part of the modern computer age—virtually all personal computers have the ability to create, store and display their content. The Internet has provided the world’s population with access to a wealth of such documents on a never-before seen scale.

However, the entire contents of a document, be it paper-based or electronic, is often too large to be usefully presented to the user at a single point in time. In paper documents, this issue is overcome by placing the text and images onto multiple sheets of paper. Electronically, this problem is almost universally solved by inserting the content into a scrolling environment. The view onto the document is then modified by directly adjusting a scrollbar or by employing tools such as the mousewheel or paging keys. Applications may also provide methods for adjusting the document’s zoom and page layout.

The widespread deployment of the scrollbar has made it the *de facto* standard for navigation in large information spaces. However, researchers do not currently have a detailed knowledge of how this and other navigation tools are used in a *real world* context. Previous work has provided motivation for such knowledge, with researchers indicating that scrolling is an “obvious case where widget design could make a difference” [37, pg. 550], that it is “irritatingly slow and distracting” [181, pg. 338] and that users would benefit from “quicker, more effortless navigation” [181, pg. 342]. A thorough characterisation of users’ actions in real-world electronic document navigation systems would significantly benefit interaction designers when developing new navigation tools. It would provide insights into common behaviours and empirical data on areas of inefficiency.

To fill this knowledge gap, this thesis aims to understand and improve desktop computer-based navigation within documents. To achieve this goal, users’ every-

day actions in two commonly deployed applications, Microsoft Word and Adobe Reader, were studied. A software tool was created to unobtrusively record these actions. It was then installed on the computers of 14 participants for 120 days to monitor their behaviour. To understand why particular interactions were observed, two further studies asked participants to perform specific navigation tasks, one of which was run as an interactive session to elicit user thoughts. The findings from these three studies are used to inform the design and evaluation of a within-document revisitation tool: The Footprints Scrollbar.

The remainder of this chapter formally defines the research goals, describes the approach for achieving these goals, provides scope for this work, presents the primary research contributions and outlines the structure of the thesis.

1.1 Problem Statement

The two primary goals of this thesis are to understand and improve navigation within documents. Current document navigation tools are potentially inefficient. However, without a full understanding of the actions users currently perform, there is no clear direction for improvement. To address this issue, the thesis sets out four goals that aim to understand and then improve current navigation tools. These goals are described below. The criteria for judging their successful completion are also noted.

1. Understand the tasks that require document navigation and the electronic document navigation tools currently available to achieve these tasks. This goal will be successful if tasks requiring document navigation are clearly identified, described and organised into high-level task-groups. Electronic document navigation tools also require identification and should be classified based on their attributes or use outcomes.
2. Create an empirical characterisation of *real world* electronic document navigation. This objective will result in a description of within-document navigation describing all facets of the actions users performed during a long-term study. These results should also allow user behaviours to be generalised or classified.

3. Understand the reasons users choose to employ particular document navigation tools. Successful completion of this goal will explain the behaviours observed in the long-term study. It will also provide insights into the limits of user knowledge of currently available navigation tools.
4. Using the knowledge gathered in the previous three goals, analyse, design and evaluate a new technique to aid a commonly performed, but currently inefficient, navigation task. Successful goal completion will result in a tool that empirically and subjectively outperforms currently available alternatives.

1.2 Research Approach

Document navigation is an activity that occurs in nearly every desktop application that manipulates content too large to fit comfortably on a user's screen. The context of this research is dedicated desktop computer-based document viewers, such as the two applications studied in detail in this work: Microsoft Word 2003 [152] and Adobe Reader 7 [5]. These applications are commonplace on millions of desktop computers and they provide an array of tools that support navigation. Understanding and improving the tools in this type of application will provide research outcomes that can be generalised to many other types of navigation system.

The literature provides a large body of work that describes tasks that require document navigation and many suggested improvements to electronic document navigation tools. This work is reviewed, but not repeated, in this thesis.

Instead, this research focuses on understanding the tools used when navigating in real-world systems and then providing suggestions for improvement. The biggest challenge in the first of these goals is in the methodology for accurately and unobtrusively observing document navigation actions over a long period of time. Techniques such as interviews and screen-recorders provide a contextually-rich understanding of user actions, but require a large amount of time to manually analyse, leading to scalability issues in large studies. To overcome this problem, an application logger that records user actions in unmodified versions of Microsoft Word and Adobe Reader is implemented. This tool is then used in a 120 day longitudinal study of real-world document navigation.

One of the primary disadvantages of application loggers is their inability to unobtrusively record user justifications for employing navigation tools. To complement the results of the longitudinal study, two task-centric studies are conducted. The first, a lab based study, asks participants to perform a series of scripted tasks. These tasks provided ideal situations for employing advanced navigation tools. The second uses interactive sessions to understand *why* particular tools are chosen to perform particular tasks. In these sessions, users complete a task and then explain to the experimenter their reasons for performing the task in the selected manner.

Having formed a good understanding of document navigation actions, the data is then used to inform the design of a tool that aids a commonly performed task: within-document revisitation. The design process consists of a series of steps that validate decisions at each stage. First, further log analysis is performed, followed by two low-level validation of concept experiments. A third experiment compares the new system, the Footprints Scrollbar, against a standard scrollbar for revisitation tasks. Finally, to complement the empirical results from the third experiment, a realistic task scenario is created, to gather subjective feedback from users.

1.3 Research Contributions

This thesis makes five primary contributions to the research knowledge in the domain of electronic document navigation. These are:

1. A review of electronic document navigation tools. To the best of the author's knowledge, no comprehensive summary of within-document navigation tools exists. This review allows other researchers to more quickly analyse the existence of current tools and opportunities for future developments.
2. The development of AppMonitor, a tool for recording user actions in unmodified Windows applications. Previously, researchers had no automated tool that could collect data pertinent to electronic document navigation, unobtrusively, in *unmodified, real world* applications.
3. An empirical characterisation of electronic document navigation. Using

AppMonitor, 14 participants' document navigation actions were recorded over a period of 120 days. This characterisation presents the properties of the documents used and analyses the following: vertical and horizontal navigation, stereotypical categories of navigators, the use of page layout tools, the use of zoom tools, navigation patterns, the use of reviewing tools, document and window management, and menu, toolbar and keyboard shortcut use.

4. Two task-centric studies that provide reasons for navigation tool selection. The first asked 37 participants to perform a series of constrained tasks in Microsoft Word and Adobe Reader. The second consisted of eight interactive sessions. Users were asked to perform tasks and then provide the experimenter with the reasons they chose particular navigation tools to aid their completion. These studies also provide insights into tool selections for particular tasks.
5. The analysis, design and evaluation of a Footprints Scrollbar that aids within-document revisitation. This tool is significantly faster and subjectively preferred over a standard scrollbar for revisitation tasks.

1.4 Structure of the Thesis

The remainder of this thesis is divided into four parts: a review of the current knowledge in the domain of document navigation and the state-of-the-art electronic document navigation tools; an empirical characterisation of electronic document navigation and two task-centric studies investigating the reasons for tool selection; the development of a Footprints Scrollbar that aids within-document revisitation; and a discussion of the research and future directions for work in this area.

To begin, Part I reviews paper-based and electronic document navigation and investigates the tools developed to support electronic within-document navigation. Chapter 2 establishes definitions of documents and document navigation in the context of this work and investigates tasks that require navigation. These tasks include 'overviewing and browsing', 'reading', 'annotating and writing',

‘searching’ and ‘revisitation’—a category of task present in all of these activities. Finally, human-centric and automated methods for observing and recording document navigation activities are reviewed.

Chapter 3 takes a more narrow focus, reviewing the tools that are available for electronic within-document navigation. Eight categories of tools are described: core navigation tools (those commonly found in today’s navigation systems), input devices, scrollbar augmentations, content-aware navigation aids, visualisations that provide multiple document views, indirect manipulation techniques, zooming tools and revisitation tools. Document navigation techniques on mobile devices are then considered and finally methods for evaluating document navigation tools are discussed.

The primary goal of Part II is to characterise the actions users perform when they are navigating within electronic documents. There are three primary components to this research: a description of the AppMonitor logging tool, an empirical characterisation of navigation actions and two studies investigating the reasons particular navigation tools are employed.

Chapter 4 describes AppMonitor, an application logger developed to automatically record document navigation actions in Microsoft Word and Adobe Reader. AppMonitor can record low level interactions such as “left mouse button pressed” and “Ctrl-f pressed” as well as high level ‘logical’ actions such as menu selections and scrollbar manipulations. AppMonitor’s architecture is described, implementation issues are discussed and lessons that will benefit other researchers creating similar logging systems are presented. A brief introduction to parsing and analysing the log files is also provided.

AppMonitor was then employed to conduct a longitudinal study of the electronic document navigation actions of 14 participants over 120 days. Chapter 5 details the results and analysis of this study. It describes the use and re-use of documents, the use of vertical and horizontal navigation tools, the percent of users’ time spent navigating and the use of zooming, page layout and window management tools. Common user behaviours, such as within document revisitation, are reported, along with a classification of users by placing them into stereotypical navigator categories.

The longitudinal study of Chapter 5 reported a large number of statistics regarding the navigation tools that were used; however, it did have one significant

shortcoming: a lack of understanding of the reasons why particular navigation tools were employed. To investigate this area, Chapter 6 describes two task-centric studies. The first, a laboratory based study, asked 37 participants to perform a constrained series of tasks to understand the tool selections that would be made. The second, a series of interactive sessions, asked participants to perform specific navigation tasks and then provide feedback to the experimenter describing the reasons they selected particular navigation tools.

Having presented a characterisation of document navigation in Part II, Part III develops a new navigation mechanism to aid one of the tasks shown to be regularly performed—within-document revisitation. Chapter 7 presents the design of a new aid for revisitation, the Footprints Scrollbar. The Footprints Scrollbar drops a small mark into the scrollbar trough whenever the user pauses on a region of the document. It then provides several mechanisms for quickly and accurately returning to these previously visited document locations. The design process for the Footprints Scrollbar begins with further log analyses to determine the optimal size and type of the required history list. Two experiments then validate the usefulness of placing marks into the trough and the degradation of revisitation performance as the number of marks increases. Next, the Footprints Scrollbar concept is presented and is validated using two further studies. The first, a controlled laboratory study, showed that the Footprints Scrollbar is significantly faster than a standard scrollbar for returning to recently visited document locations. The second, a realistic usage study, gathered subjective opinions of the system. Participant responses were overwhelmingly positive, many asking when they could have it incorporated into their own document viewers.

Part IV provides a discussion of the research presented in the thesis and documents future directions for work in this area.

Part I

Document Navigation and Document Navigation Tools

Chapter 2

An Overview of Paper and Electronic Document Navigation

DOCUMENTS are diversely used: an enthusiastic child will read a novel from cover to cover; an irate consumer will search a product manual when a device fails to function. These activities require that the user have access to all of the document content. However, this content is often too large to be viewed in its entirety, at a single point in time. To overcome this issue, documents are published in a manner that facilitates temporal viewing—the pages in a book are viewed one-by-one and a scrollbar controls the region of a document that is currently shown on-screen. These actions are examples of document navigation. They are essential for all but the shortest of documents or the simplest of tasks.

Current knowledge in the domain of document navigation is reviewed in this chapter. First, definitions of a document and document navigation are presented, along with by a breakdown and description of the sub-tasks involved in a single document navigation action. Second, five broad categories of user tasks requiring navigation are outlined, based on prior document navigation characterisations. These categories are ‘overviewing and browsing’, ‘reading’, ‘writing and annotation’, ‘search’, and ‘revisitation’. Finally, human-centric and computer-centric methods of observing and recording document navigation are described.

2.1 *Foundations*

The Concise Oxford Dictionary defines a document as “a piece of written, printed, or electronic matter that provides information or evidence or that serves as an official record” [185, pg. 421]. This thesis focuses on navigation in a subset of these “documents”: desktop computer-based electronic documents. Desktop-

based documents are those that are stored and manipulated on a user's computer (as opposed to those provided by, or manipulated on, a remote server). These documents are composed of words, sentences and paragraphs that are divided into pages. They may also include images, tables, figures and diagrams. Two typical examples are academic conference papers and instruction manuals. For the purposes of this research, the thesis defines a document as:

Document: *A piece of electronic matter stored and manipulated on a desktop computer. It provides information or evidence or serves as an official record. It is composed primarily of words that make sentences and paragraphs and may include diagrams, tables and images. The content is arranged into pages.*

However, this research does not exclude other forms of electronic documents such as web-pages, program code, photo libraries and spreadsheets. Many of the navigation tools and techniques the thesis describes can equally be applied to this broader category of documents.

Having established a definition of a document for this context, it is necessary to also define *document navigation*. A user will move within a document in order to achieve a specific task, for example, reading the document from start to end, moving to a specific page or searching for a particular phrase. Each task may require the user to adjust the position or orientation of either their head, or more often, the document. Within-document navigation can therefore be defined as:

Within-Document Navigation: *Adjusting the view of an active document to achieve a particular task.*

The *view* onto a document describes both the position within the document and the representation of that position to the user. The definition of *active* is medium dependent. For electronic documents, an active document is open (so it does not include file-management tasks); for physical documents, it is the document currently under inspection.

In contrast, *between* document navigation involves moving from one document to another. Typical examples of this include returning a book to a shelf and picking up another, or following a hyperlink from one web-page to the next¹.

¹Note that hyperlinks also support navigation *within* the same document.

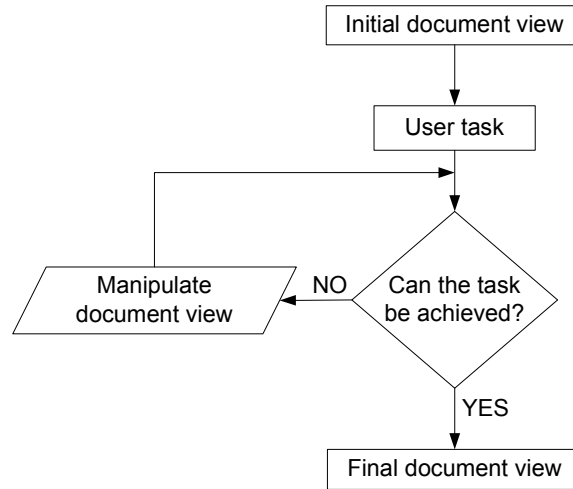


Figure 2.1: Sub-tasks in a document navigation action.

Throughout this thesis, unless otherwise stated, the term *document navigation* refers to *within* document navigation, as described here.

2.1.1 Document Navigation Sub-tasks

Each document navigation action consists of a series of sub-tasks, as summarised in Figure 2.1. The document will begin and end with a particular view. *View* is medium-dependent, but it will at least define the position, orientation, page layout, and closeness or zoom of the document.

The user will have a task to achieve that may require a modification of the view onto the document. User tasks are considered at the lowest possible level. For example, the task of “determining if this paper is worth reading” may involve reading the abstract, skimming over the majority of the paper and reading the conclusion. This task will require several navigation actions to achieve: the user will first move to the abstract, then slowly move the document for comfortable reading of the abstract, and so forth.

The user first checks to see if they can achieve the required task with the current document view. If not, the document view is adjusted. Again, the document view is queried to determine if the task can be achieved. These last two steps are repeated until the user is satisfied with the document view or can achieve the

required task. The final document view is then reached.

The user task impacts on the type of navigation and the mechanisms used to achieve the required task. The task of ‘slowly moving a document so it can be read’ requires a different type of navigation to that used when moving to a specific position within the document. The following section investigates user tasks that require document navigation, Chapter 3 details tools available to manipulate the document’s view.

2.2 Navigation Tasks Within Paper and Electronic Documents

Prior to the modern computer age, paper was the medium of choice for virtually all documents. Even today, vast libraries of paper-based documents and books are commonplace. Many documents are published on paper and electronically so that the advantages of each medium can be utilised. The prevalence and longevity of paper-based documents provide context for the actions readers may wish to perform in their electronic counterparts.

Navigation tasks in paper and electronic documents are examined in this section. First, ‘overviewing and browsing’ a document is described. Second, different styles of reading are investigated, followed by annotation and writing tasks. Finally, document searching and document revisitation tasks are presented.

2.2.1 Overviewing and Browsing a Document

Overviewing and browsing allows the reader to obtain an idea of the content of the document (text, titles, pictures, maps, diagrams), the size of the document (short, medium, long) and their interest level in the topic. Consider a shopper who has picked up a book off a shelf. The shopper will read the endorsements on the back of the book to judge its quality and use its tangible aspects (width, height, breadth and weight) to roughly calculate its length. The shopper will then browse the book by quickly flicking through the pages, gaining an overview of the content. A decision to purchase the book, place it back on the shelf, or begin to scan or read parts of the text will follow. In this situation, the shopper used the physical pages and tangible aspects of the book to support the overviewing and browsing tasks.

Readers who are familiar with the *structure* of a document perform overviewing and browsing actions to acquaint themselves with new content. Marshall and

Bly's [144] field study of magazine readers observed all participants used these techniques after receiving the most recent copy of a magazine subscription. The first participant always began by looking over the table of contents or by quickly leafing through the magazine. The second participant "sets his expectations" [144, pg. 229] by viewing the table of contents and then began linear reading from start to end. The final participant looked at the table of contents and then flipped through the magazine until interesting articles were located.

Loose-leaf documents are less amenable to the page riffling or flipping available in magazines and books, but instead allow flexibility of page layout for overiewing. O'Hara and Sellen's [181] laboratory study of reading paper and electronic documents found that participants used the spatial layout of loose-leaf sheets on their desk to gain a sense of the overall structure, to cross-reference and to interleave reading and writing. All participants visualised multiple pages simultaneously. When using the electronic documents, participants were hampered by limited screen real-estate, meaning multiple pages could not be displayed at reading zoom. This study was conducted in 1997—improvements in screen technology since this time mean multiple pages can now be displayed at reading zoom, especially on wide-screen monitors.

A document's length can influence the decision as to whether to continue reading—long documents may be too time-consuming to complete, while short documents may not provide sufficient detail. Physical documents allow a reader to judge both a document's length and their position within the document. This information can be determined implicitly as pages are turned. Electronic documents provide page counters and widgets such as the scrollbar to represent document position. However, these mechanisms require explicit attention—users who are unaware of these functions may be forced to scroll through the complete document [181].

Document triage is the process of determining the relevance of a document and whether it warrants further inspection. Buchanan and Loizides [32] describe a user study of document triage in paper and electronic documents. In the electronic condition, "many documents (34%) were never scrolled, and 64% not read beyond the first page" [32, pg. 422]. Participants spent 68% of the time on the first page of the PDF, without scrolling, the remainder was split between scrolling activities (15%) and stationary reading of content (17%). In the paper condition, 47% of

time was spent on the first page, with the remainder spent reading (nearly 50%) and within-document navigation (<5%). The authors make two further relevant observations: skim reading is more likely in electronic documents and there was a “declining likelihood of text being read when it fell later in the document” [32, pg. 422]. In a study of document relevance, Kelly and Belkin [122] reported that users scrolled an average of 4.28 times per document. Scroll actions were recorded as “the number of times the user clicked on the scrollbar” [122, pg. 408].

2.2.2 *Reading*

Documents are produced as a record of information, for later editing or reading by interested and permitted parties. The reading audience may range from a single person (for personal notes) to hundreds, thousands or millions of people if the document is posted in a publicly accessible place, such as in a library or on the Internet. The vast range of documents created for numerous audiences mean that a large number of reading strategies will be employed depending on the person, purpose, situation and document presentation.

Reading is the most common of document-related activities. Adler et al. [3] studied a week of the work related reading tasks of 15 people in a variety of professions. The authors report that reading occurs in 70% of document-related activities. Marshall and Bly [144] back this up, observing 70–84% of magazine readers’ time was spent reading (and the remainder on tasks such as scanning).

With reading accounting for a large portion of document-related activities, navigation actions should be efficient and unobtrusive to avoid interrupting the reader during this activity. To aid this goal, it is important to understand both *how* a document is read (fast or slow, linearly or non-linearly), and *why* it is read. These two factors influence how the document will be navigated. This section addresses these issues and ends with a comparison of the differences of reading from paper and reading from a screen.

How Documents are Read

The manner in which documents are read impacts on the navigation behaviour of the user. Lunzer [138] suggests four different types of reading: receptive reading, reflective reading, skim reading and scanning. Each type of reading demands

different attention levels and variations in linearity.

Receptive reading: Is likened to listening to the text read aloud. The reader will comprehend the content by relating recently read information to that just read. This activity is supported by adjusting the rate of reading over time.

Reflective reading: Reading that is “frequently interrupted by moments of reflection” [138, pg. 26]. The reader uses this time to consider the content and meaning of the document.

Skim reading: Rapid reading to obtain a rough overview of the document content. This style of reading is often used to determine whether receptive and/or reflective reading should take place.

Scanning: Similar to skim reading, but with the aim of determining whether the text contains a particular piece of information; if this is already known, to relocate it (this is similar to visual search, see Section 2.2.4).

The order the text is read heavily influences how it will be navigated. In the case of electronic documents, the reading order also impacts on the required tools. O’Hara [180] notes two important distinctions:

Serial/Non-serial reading: The difference between reading a text in a linear fashion (serial) or moving between sections of text in a non-linear manner (non-serial).

Single/Repeated reading: Whether the text is read only once or multiple times. Repeated reading may occur in the same session or across multiple reading sessions.

The Purpose of Reading

There are a wide variety of reasons *why* a document is read, all of which influence *how* it is read (as described in the previous section). O’Hara [180] provides a thorough summary of typical reading goals. These goals are briefly outlined below, interested readers should see his work for a more complete review of the literature. Each goal is supplemented with the types of reading likely to be associated with

that activity (shown in italics). Some of these reading types are explicitly stated by O'Hara, others are implied.

Reading to learn: Reading in order “to be able to answer questions at a later date” [180, pg. 8]. Generally, this type of reading is slow, with a high attention to detail. Initial surveys of the material may involve skim reading.
⇒ *Skim, receptive, reflective and repeated reading.*

Reading to self inform: Reading that takes place to fulfill a desire for further knowledge on a topic area, “without any specific task or goal to which it will be applied” [180, pg. 8].
⇒ *Receptive and reflective reading.*

Reading to search/answer questions: Searches may be clearly- or ill-defined with a varying amount of information required for a suitable answer. Searching is influenced by document length, time constraints, certainty of locating relevant content and the required output precision.
⇒ *Scanning.*

Reading for research: Research involves distilling information from multiple documents on a particular topic, in a given timeframe. For each document, the researcher first decides whether to continue reading the document (by skimming), if so, a detailed linear reading or non-linear scanning of interesting sections is then performed.
⇒ *Skim, scan, receptive and reflective reading.*

Reading to summarise: The reader aims to “extract ... a general theme as well as the important and interesting points” [180, pg. 10]. This task directly involves writing while reading.
⇒ *Receptive reading.*

Reading for discussion: Reading to prepare for a class or meeting. The reader is generally interested in high level main points, not low level details.
⇒ *Skim reading.*

Proof reading: Involves “identifying errors at a grammatical and syntactic level” [180, pg. 10] and recording or annotating these errors. There is little need for moving between different parts of the document, as proof-reading occurs at the sentence level. It is often performed in conjunction with *text revision* and *critical review*.

⇒ *Serial and receptive reading*.

Reading while writing from multiple sources: Documents are analysed and evaluated by comparing the content and consistency across multiple sources. This task is often part of the research process and is supported by note-taking.

⇒ *Reflective, scan and non-serial reading*.

Reading for text revision: Often performed in tandem with *proof-reading*. The reviser (either the author, or a second party) will check for spelling and grammatical errors (proof-reading), evaluate the writer’s intentions, and assess the structure, interestingness and completeness.

⇒ *Serial, repeated, receptive and reflective reading*.

Reading for critical review: This is closely related to *text revision*, but for someone else’s document. A reviewer needs to read the document to assess: writing clarity and precision, grammatical correctness, overall presentation and the content that describes the author’s work and ideas. One example of this type of reading is the work performed by reviewers of academic journals.

⇒ *Receptive, reflective, scanning, repeated and non-serial reading*.

Reading to apply: The type of reading performed when the text contains orders or instructions on how to achieve a goal (for example, instructions on how to assemble a piece of furniture).

⇒ *Receptive and repeated reading*.

Reading for problem solving and decision making: Reading in order to make a decision on a problem. Often this requires the reader to “access and read information from multiple and disparate sources” [180, pg. 14].

⇒ *Receptive, skim and non-serial reading*.

Reading for enjoyment: Can either have high emotional involvement in the text, such as when reading novels, or it can be lighter such as letters, comics or magazine articles. This type of reading is generally linear.
⇒ *Serial and single reading.*

Adler et al. [3] categorised the work related reading tasks of participants from a variety of professions. Reading to search/answer questions, reading for cross-referencing and reading to support discussion were the three most common reading types—each accounting for over 20% of readers’ time. These were followed by skimming, editing or critically reviewing, and reading to self-inform.

The reading goals described above are generally document and media independent. However, the presentation medium influences the manner in which the reading will take place and the techniques used for navigation. The differences between reading from paper and reading from a screen are now discussed.

Comparing Reading from Paper and Reading from a Screen

Dillon [62] provides a wide-ranging review of the literature that compares reading from paper to reading from screens prior to 1992. The author identified two core areas of performance evaluation between the two media: *outcome* and *process*.

There are five areas of evaluation for the reading outcome: speed, accuracy, fatigue, comprehension and preference. Dillon’s survey found that reading is significantly slower from screen than paper, but the accuracy between media is not significantly different (although cognitively more demanding tasks can take longer on screen). Fatigue is not inherent in the task of reading from screens, although performance may be more difficult to sustain over time. Dillon’s review suggests that a good quality screen can reduce fatigue. Comprehension is not negatively affected by reading from a screen, however the speed with which a given comprehension level is attained may increase. Finally, preferences were shifting from paper to screen as the quality of displays improved.

Measures of the process of reading using the two media include: eye movements, manipulation and navigation. There were not “gross differences” observed when comparing eye movements between screen and paper. However, electronic texts are seen as the “less manipulable” medium. Dillon indicates a consensus between researchers that navigation is “the single greatest difficulty for readers of

electronic text” [62, pg. 1307] and it is an area that is worthy of further research.

Dillon’s literature review also notes attempts to isolate the root cause of differences between paper- and screen-based reading. These causes include: orientation (paper is flexible, screens are not), visual angle (screens tend to have longer lines, so readers adjust their positions), aspect ratio (paper is typically higher than it is wide, whereas screens are the opposite), dynamics (screen filling style, screen filling rates and the direction of scrolled text), flicker, image polarity, display characteristics (character size, line spacing and character spacing), anti-aliasing and user characteristics. One or more of these properties contribute to the outcome and process differences described earlier.

Since Dillon’s review, advances in technology have addressed some of the issues noted for causing differences. For example, LCD screens do not flicker and are more amenable to rotation to match the traditional orientation of paper-based documents. Technologies such as *ClearType* [158] can increase reading speed and visual search speed [63].

The physical nature of paper can benefit magazine reading and text summarisation tasks. Marshall and Bly’s [144] study of magazine readers observed *lightweight*, unselfconscious navigation to be a crucial factor missing when reading electronic documents. Four categories of lightweight navigation were observed: narrowing or broadening focus by manipulating physical pages, eye stray to page elements out of the text flow, looking ahead to preview or anticipate, and looking back for re-reading. Narrowing and broadening focus was achieved by moving the magazine so the top or bottom of a page was in view or by physically folding the magazine so only a single column was visible. Similar manipulations of an electronic document would likely require more effort, using a series of zooming and window resizing actions. Eye stray and look ahead can occur for two different reasons: a prominent feature may draw the reader’s eyes or the reader may have interest in the content that follows. One reader in their study found large cartoons distracting—they had to attend to the cartoon before reading of the main article could continue.

O’Hara and Sellen’s [181] comparison of paper and electronic reading also found the physical nature of paper to be beneficial to readers. Movement within and between documents, spatial layout and annotation while reading were noted as tasks generally easier with paper than on screen. Paper navigation was charac-

terised by its “speed and automaticity” [181, pg. 338], with the use of two hands and partial page turning before reading was completed. In comparison, navigation in electronic documents was found to be “irritatingly slow and distracting” [181, pg. 338] with single handed input meaning navigation had to be performed serially with other activities. Spatially laying out the document on a table was used to gain a sense of overall structure (as described in Section 2.2.1), to cross-reference and to interleave the tasks of reading and writing.

Finally, Marshall and Bly’s study characterised within-document navigation in ePeriodicals and in their paper counterparts. The authors reported “rough” statistics on movement in the documents: 76% of electronic and 72% of paper actions were forward movements, 8% and 13% were backward. Jumps accounted for 16% of ePeriodical movement and 6% of paper movement. Page flipping was observed in paper documents for 9% of the movement, but it was not observed at all in the electronic equivalent.

2.2.3 Annotating and Writing Documents

Annotation, note-taking and writing require the composition of new words or markings on an existing or partially completed document. Annotation and note-taking occur in conjunction with reading—the notes and markings made during these activities will aid the completion of another task. For example, when reading to learn, important document sections are highlighted and notes made to instill important information; when proof-reading, mistakes are annotated on and around the incorrect content. Note-taking can also aid the process of composing a new document. Document creation involves a series of phases including planning, writing and revising.

Adler et al. [3] examined how these writing based tasks were divided between different activities by studying the work-related writing of 15 participants, with a variety of vocations. Completing forms accounted for 33.9% of activities, annotation 26.4%, note-taking 21.7%, creation 16.4% and updating 1.6%. These tasks are described in this section.

Annotation

Document annotation is a distinct type of writing, where the annotator places marks on top of pre-existing text. Annotating a document while reading deepens the reader's understanding of the text by marking pertinent areas, it aids planning when writing a summary and is used to highlight proofing errors. Proof-reading requires the identification of "errors at a grammatical and syntactic level" [180, pg. 10], from spelling mistakes through to identifying unclear, confusing or misplaced passages. It is performed by the original author (where it is considered part of the writing process) or by one or more secondary parties. When performed by a person who is not the original author, errors are corrected or annotated directly on the document. Researchers and document reviewers use annotation as a time-efficient and integrated method of engaging with the text, by recording comments, references and noting important material [182].

Significant differences exist between paper and electronic annotation. Paper annotations are "relatively effortless and smoothly integrated with reading" [181, pg. 337]. They allow references to pre-existing text using pointers and indexes that are easily distinguished from the original document's content [182]. Readers are more likely to make annotations on printed documents, than on their electronic counterparts [181].

In contrast, annotation in electronic documents is considered "cumbersome" [181, pg. 337] and detracts from the reading task. The inflexibility of input using the mouse and keyboard make annotation hard. O'Hara et al.'s [182] study of researchers using traditional libraries found that shifting from paper-based to on-line note-taking makes it more difficult to create the "free-form and idiosyncratic style of paraphrased information and reader comments that might normally be interleaved with verbatim information" [182, pg. 239]. Users may also be cautious about destroying the original document's integrity—annotations are perceived as "a separate layer of the document" [181, pg. 337], but this is difficult to achieve in an electronic setting [181].

The gap between the ease of making paper and electronic annotations has resulted in the development of several media-bridging systems. The *Paper User Interface* [114] scans annotated cover sheets to determine the actions to take with the accompanying paper document, be it storage or distribution by fax. *Paper-*

Link [17] is a hybrid system that allows marks made on paper to be “associated with electronic content” and for words on the paper to be used as input to an electronic system. In a similar vain, *Digital Paper Bookmarks* [208] are adhesive stickers that can be attached to physical documents and are later synchronised to become electronic bookmarks.

How Documents are Written

To understand *how* documents are written, researchers have devised models to describe the processes involved. These processes provide insight into the navigation mechanisms required to support the task of writing. The literature reveals two schools of thought on the writing process. The first is a linear model where the writer moves through a series of stages to achieve a product. The second is a recursive or cyclic model that focuses on re-iteration of various steps that may occur in a non-linear fashion. These models, along with Hunter and Begoray’s [105] encompassing framework, are described in this section.

Linear models of writing describe a series of discrete steps that the author follows, from an idea through to a completed work. Rohman [194] viewed the writing process as three distinct phases: pre-writing, writing and re-writing. Pre-writing is the “stage before words emerge on the paper” [75, pg. 367], writing is the production stage and re-writing is the process of re-working the output from the previous stage [75]. Collins and Gentner [56] viewed writing as “a process of generating and editing text within a variety of constraints” [56, pg. 52], namely structure, content and purpose. Their framework separated *idea production* from *text production*, arguing that text production produces a linear sequence, while idea production is a “set of ideas with many internal connections, only a few of which may fit the linear model” [56, pg. 53].

Opponents of the linear model argue that it inaccurately describes the writing process. Flower and Hayes [75] instead describe writing using a cognitive process theory. Their model describes the three major elements of writing: the task environment, the writer’s long-term memory and the writing processes: planning (generating, organising and goal setting), translating, and reviewing (evaluating and revising).

Many of these models are similar, with only minor differences in the division

of tasks. However, there are more significant differences in how these tasks interact. Hunter and Begoray [105] created a framework that aimed to encompass the commonalities of these models. Writing was divided into four activities: generating, organising, composing and revising. Hunter and Begoray do not suggest the order or frequency that these activities occur, as the literature contains opposing views as to the linearity or recursiveness of these tasks.

Generating is the process of deciding on the content, gathering one's thoughts and collecting information from one's own memory and external sources. The information is then organised, with irrelevant pieces discarded. The *organising* process involves creating a document outline (be it linear or hierarchical) that results in ideas, plans, notes and thoughts being gathered and translated into written form. *Composing* is the translation of this initial outline into sentences, paragraphs and a 'draft' document. The act of *revising* involves reading the composition and altering the content by insertion or deletion of text, reordering of words, phrases or larger units of text, substitution of units of text to include new ideas or adjusting the material to better fit their original intentions and increase comprehensibility.

Why Documents are Written

Understanding *why* a document is written provides insights into the form it will take and the types of navigation it may require. Writing is one of human-kind's primary forms of communication and there are numerous reasons for creating document-based communication. MacArthur et al. [139] reviews many of these reasons:

Personal communication: Writing for those "who are removed by distance or time, allowing us to maintain personal links with family, friends and colleagues" [139, pg. 1], this may be immediately relevant (such as writing an email) or persist, to preserve history and heritage among groups.

Persuasion: To re-enforce or alter opinions based on a compelling argument; for example, political or environmental statements or 'letters to the editor'.

Convey knowledge and ideas: The purpose of news media, but also common in academic journals and conference papers.

Maintain (civil) order: Laws that govern what can and cannot be done.

Administration: Business, social and political organisations have a standard set of operating protocols that govern the administration of the group. These include documented output in the form of memos and minutes from meetings.

To apply: Job applicants must complete application forms (as well as writing to persuade the employer they are the right person for the job).

To teach: Instruction manuals describe new skills, the operation of devices and procedures for construction. These are produced to teach the reader how to perform a particular task.

Personal expression: Is used by authors to “explore who they are, to combat loneliness, to chronicle their experiences, and to create alternative realities” [139, pg. 1]. This work may remain personal or be published in books or magazines, or on the Internet.

2.2.4 *Searching Within A Document*

Search techniques are used to locate information within a document. The three main search techniques are: *visual search*, *goto* and *find*. Visual search employs the user’s perceptual system to scan each page of the document, until the target information is located. Goto uses pre-acquired knowledge of the document or indexing features to move directly to the desired document position. Find is a hybrid of visual search and goto functionality—users may either have a “rough idea” of the desired document location or they may employ tools to aid in locating document positions that met a certain search criteria. Factors that determine when each tactic is employed include: the type of information required, the precision of the search and the familiarity of the user with the document.

This section begins with a short discussion of between-document search engines and their relevance to within-document search. It is followed by descriptions of visual search, goto and find.

Search Engines

Internet-based search engines such as Google [84], Yahoo! [235] and Microsoft's Live Search [151] allow users to quickly scour vast numbers of web-based documents. *Between* document search is not the focus of this section or indeed this thesis, however within-document search tasks can be performed using search engines. In their studies of information seeking, Loizides and Buchanan [137] reported on a participant who never used specifically designed within-document search features, such as the Ctrl-f tool, but instead made use of the flexibility of online search engines. Google, for example, supports exact and inexact phrase matches, 'OR' expressions, exclusion of pages by words they contain and filtering by date, usage rights or the location of the search terms within the page. Suggestions are also offered when words are spelled incorrectly. Desktop-based search engines, such as Windows Vista Search [161], Google Desktop [82] and Spotlight for Mac OS X [13] also allow users to search for files based on their name, modification date or contents.

Visual Search

Visual search uses the human perceptual system to locate and identify a predetermined object or document chunk, such as a paragraph of text. Visual search is employed when the object under consideration is either not exactly known (for example, "the section that describes the assembly") or not accessible more efficiently (for example, "the map of the world" in a document with no figure index). In some instances, the user is the only person who can identify the correct position in the document, for example "the paragraph below the informative diagram". This section briefly outlines significant models of visual search and factors that can influence the speed and accuracy of object location.

There is a large body of literature describing attributes and models of the cognitive processes involved in visual search. Different types of object attributes are recognised at different stages in the visual search process. Wolfe [230] summarises the research into two classes of attributes: basic attributes and conjunctive attributes. Basic attributes are singly differentiable, for example a single red dot amongst green ones. Basic attributes include colour, orientation, curvature (of lines), vernier offset, size, spatial frequency and scale, motion, shape, pictorial

depth cues, stereoscopic depth and gloss. Conjunctive attributes are unions of multiple basic attributes, for example “a red vertical line in a background of horizontal red and vertical green lines” [219, pg. 99].

The *Feature Integration Model* [219] describes a two stage model of visual search. First a parallel search takes place for basic attributes, if that fails a serial phase of searching takes place for targets that are conjunctions. The now widely accepted improvement to this model is the *Guided Search Model* [231]. The Guided Search Model identifies that the parallel and serial searches are not completely independent and that the parallel search can guide the final serial search stage, eliminating the need for some repetition.

Visual search is not always successful. The required object will not always be identified, either because it was missed or because it is not contained within the document.

Goto

Goto searches require the user to have a precise knowledge of their target within the document. In paper documents this is usually in the form of a page number. Users may acquire this knowledge from external sources (such as page reference from another document), from tables of contents or from indexes contained within the document. Alternatively, the conventions of the document may allow a goto search. For example, the user may know that the endorsements are always printed on the back cover of a book.

Find

Find is a hybrid of visual search and goto techniques. There are two general cases of within-document find. In the first, the user will have precise knowledge of the target, however the target may still be ambiguous. A typical example is within-document *phrase search*. When using an electronic document the user enters the phrase of interest. The application lists or cycles through all occurrences of that phrase. In paper documents this task must be performed manually, using scanning, visual search and reading. Users report that this type of find functionality is one of the key advantages of electronic documents over their paper counterparts [32].

The second type of find occurs when the user has knowledge of the approximate position of the target within the document. For example, the user might know “the methodology section is about a third of the way through”. The user will perform a goto action to the approximate position. Visual search and reading strategies are then employed to precisely locate the target.

2.2.5 Document Revisitation

Revisitation is a common activity in many aspects of computer use. Greenberg and Witten studied the reuse of Unix-based command lines, finding “an average of three out of every four command lines entered by the user already exist on the history list” [87, pg. 364] and that *temporal* recency contributes heavily to reuse. Internet revisitation in web browsers is widely studied. Revisitation rates (the probability of a page visit being a revisit) range from 43.7% [179] to 81% [49], with a significant number of these revisits occurring soon after the original access [2, 179, 214]. Obendorf et al. [179] reports on the tools used for returning to these pages. The ‘back’ button was used for 31% of all page revisits, while the bookmarks, homepage button, the history list and typed URLs were only responsible for 13.2% of page revisits. Finally, menu revisitation evaluations show user-adaptable menus to be faster than system-adaptive [73] and that familiarity with menu items can be accurately modelled [54].

Within-document revisitation occurs when a user returns to a document position they have previously viewed. Several studies report this type of navigation to occur in both paper and electronic documents. Marshall and Bly [144] observed magazine readers going back in the document to re-read part of an article, to refresh details or double-check material. O’Hara [180] also notes that re-reading is an important aspect in “reading to learn” tasks.

The “fixity of information” [181, pg. 338] on pages in physical documents means readers acquire incidental knowledge of the spatial location of the document’s content [181]. The constant location of content is beneficial to users; recalling the on-page position of an object significantly reduces the search area for revisitation tasks. Scrolling an electronic document removes some of this fixity—the relative position of the content remains the same, but the absolute position is continually changing. For example, the content that is in the top right corner of

the screen changes with every scroll action, unless whole pages or book layouts are used.

In paper documents, *dog-ears* can be used as semi-permanent markers of places, ideas, or even particular words [195]. To create a dog-ear, the user folds down the corner of a page. This marker will remain until the reader unfolds the page to return it to the original state. Temporary markers are created when the reader uses their fingers as a location aid [62].

2.3 Observing and Recording Document Navigation

An important preliminary to developing new document navigation techniques is an understanding of how current systems are used by their target audience, in their everyday environment. In the case of electronic document navigation this target audience is vast, with skill levels ranging from complete beginner to expert. The previous section described observations and recordings of document navigation tasks that already exist.

The tools available to the researcher to carry out these observations and recordings are explored in this section. The techniques fall into two broad categories: human-centric observations and automated logging techniques. Human-centric observations provide good contextual information, but less accurate empirical data. Automated logging techniques allow recording of extremely accurate empirical data, but usually capture less contextual information. Each group of techniques has its own set of advantages and disadvantages which are also discussed.

2.3.1 Human-Centric Observation

Human-centric observation heavily involves the researcher in the data collection and/or the data analysis stages. Researchers employ techniques such as video analysis and transcribing along with think-aloud protocols to elicit user thought patterns. Participants will actively or passively aid data collection by simply completing tasks (passive) or providing their thoughts and opinions (active). Human-centric observation has two primary disadvantages: the large amount of experimenter time required for data collection (limiting the breadth of studies) and the possible implications of the Hawthorne effect (where observation influences behaviour) [147].

Researcher Involvement

The researcher will passively or actively participate in the data collection process. Passive data collection is purely observational. The researcher will take notes and make audio and/or video recordings of the participant's behaviour. The recording(s) and notes are later reviewed to draw conclusions from the observation. Active data collection uses techniques such think-aloud protocols [132, pgs. 83–86], pre- and post-hoc surveys, and semi-structured interviews [77] to extract user thoughts and experiences.

Think-aloud protocols require the user to describe the actions they are taking and possibly the reasons for selecting those actions. The researcher can encourage this process by reminding the user to explain their thoughts when the commentary runs dry and by asking non-leading questions, for example “tell me what you are thinking” instead of “what do you think those prompts ... mean” [132, pg. 85]. In active studies, the researcher may also offer the participant aid if they get stuck. During passive studies, participants are left to solve all problems themselves—this in itself gives interesting insights into the usability of a system.

Surveys provide a formal, structured method for users to report on specific aspects of the study and often also encourage freeform comments. Semi-structured interviews are based on a set of core questions that the interviewer uses to probe certain areas of interest. These are more flexible than surveys and allow impromptu follow-up questions to provide deeper insights into user thoughts.

To ease their workload during a study, researchers may set up audio and video recordings to compliment their notes. These recordings are used as a reference for situations where the researcher's notes are incomplete or as a full record of the observation. In certain situations, the researcher may not be present during the observation—in these cases, a video camera is set up to record all of the participant's actions. When a full record of events is required, the audio or video sources are transcribed. Transcription involves creating a complete and accurate written record of everything spoken and possibly all of the participant and researcher actions. Software packages such as *Express Scribe* [176] or *Hyper-TRANSCRIBE* [191] can aid this process.

Participant Involvement

Participants are also either passively or actively involved in providing the researcher with information. When passive, the user simply completes the tasks required by the researcher without explicitly providing feedback—the researcher will gather this from the user’s actions. Active involvement requires the user to provide feedback on their experiences. This includes think-aloud protocols, surveys and all types of interviews. Different parts of a study may require both passive and active involvement from the participant.

Finally, participants can be asked to observe themselves using diary studies and self-reporting. In these methods, the participant regularly notes the activity they are performing at a certain time or if the action under study is rare, the date and location they performed it. This method of reporting heavily relies on the conscientiousness and consistency of the participants and is easily open bias [192].

Advantages

The two primary advantages of human-centric observation are the large amount of contextual information available and the flexibility of the observations. Researchers can accurately observe the context of actions and mistakes and whether certain conditions or a series of actions lead to mis-understandings. They can also quickly and accurately record both computer and environment state, particularly if these may influence the outcome of the study. The ability to probe problem areas by asking new or different questions or by using examples allows deep insights into user thoughts and actions. This type of insight is not readily available in strictly structured or automatic systems.

Disadvantages

The two main disadvantages of human-observation techniques are the large amount of experimenter time required and the possible influence of the Hawthorne Effect. When directly observing participants, especially on a one-to-one basis, experimenters must be physically present at the sessions. Further, creating transcriptions of audio and video recordings is a long and tedious process. Secondly, humans may be influenced by the Hawthorne Effect, where the behaviour of participants changes when they are under study. This effect was first reported by Mayo [147],

who observed that factory worker performance increased, in unfavourable conditions, because the participants knew they were under study.

2.3.2 Automated Logging Software

Automated logging software collects data on user interaction with applications on their computer. This is useful for short controlled experiments and longitudinal studies of days, weeks, months or even years. Logging software overcomes the two biggest disadvantages of direct observational techniques: it allows the efficient collection of a large amount of data and reduces the likelihood of participants modifying their behaviour while under study. This section describes four types of automated logging software: mouse and keyboard loggers, macro based recorders, screen recorders and dedicated application loggers.

Mouse and Keyboard Loggers

Mouse and keyboard loggers record the events from their respective hardware devices using either hardware or software based techniques. Hardware loggers require installation of an extra component between the device and the user's machine. These loggers are better suited to keyboards than mice. Mice are likely to have software transformations (in the form of acceleration) performed by the operating system, meaning a 1:1 mapping between hardware mouse position and the cursor on-screen is unlikely. Keyboards are less likely to have any deviation between the keys pressed and the characters an application receives. Keyboard hardware loggers, such as *KeyGhost Keylogger* [124] and *The KeeLogger* [119] are placed between the keyboard cable and the jack on the computer. This requires physical access to the computer and involves a cost in obtaining the required hardware.

Software loggers are easily distributed, but incur their cost in development time. Many research loggers are available for a wide range of operating systems, including: Datalogger [226] for Windows 3.1 and DOS, InputLogger [220] for the Apple Macintosh, and RUI [223] for Windows and Mac OS X. These applications provide timing logs for key strokes, mouse clicks and mouse moves across all programs on the operating system. They do not provide information regarding the semantics of the application or the user's actions, meaning contextual information

such as buttons that are pressed or the state of the target application cannot be determined.

Mouse and keyboard loggers often receive negative publicity due to their use in malicious viruses and trojans to capture passwords. Researchers must be careful to overcome user apprehensions about installing such a system. One approach is to allow all recorded content to be made available to the study participant.

Macro-based Recorders

Macro recorders are mouse and keyboard loggers that additionally log semantic application information to enable automatic replication of repetitive, manual tasks. Many such commercial systems exist, for example Iolo Technologies' *Macro Magic* [110], Tethys Solutions' *Workspace Macro* [216], CprinGold Software's *Smack* [58] and Jitbit Software's *Macro Recorder* [113].

These recorders maintain knowledge of the application and its position on-screen, as well as files it had open, to allow key-presses and mouse movement to be repeated. Most allow the user to edit the macro, presenting the actions performed in a human-readable manner. The semantics of key presses or mouse clicks are often also recorded, indicating which menu item was selected or button pressed. While intended to allow commonly performed operations to be quickly repeated, these systems could potentially be used to track the actions of a user over time.

Microsoft Word's [152] built-in macro system allows the user to record a set of actions that need to be repeated. This system also allows one to redefine the functions performed when a menu selection is made or a button is pressed. This feature was employed by Linton et al. [134] in their Organisation-Wide Learning (OWL) system, to log high level commands issued in Microsoft Word 6.0 for Macintosh.

Screen Recorders

Screen recorders replicate the visual state of a user's screen or window to a video stream. Supplementing these recordings with semantic information from low level logs, such as mouse clicks, can provide emphasis on these actions during playback. One commercial example of a screen recorder system is Techsmith's Cam-

tamsia Studio [215]. The main advantage of screen recorders is that they provide a contextually rich understanding of the user's actions: the researcher can visually identify reasons for user interactions.

There are three disadvantages of screen recorders. First, they are resource intensive, consuming large quantities of processing power and storage space. Study participants will not be prepared to accept a significant negative effect on computer performance over long periods of time. Second, they do not record information on logical interface events, rather, the screen state and the position of mouse events and keyboard input. Finally, this technique scales poorly. Researchers must manually watch, interpret and record actions from the video (as in video analysis techniques used in human-centric observation), making it impracticable for large, long term studies. Retrieving empirical data on actions from the video is tedious, having to match mouse click logs with actions on-screen.

Application Loggers

The final category of automated logging techniques are dedicated application loggers. These target specific applications of interest, logging actions by modifying the base application code or by external monitoring.

Studies of web navigation have implemented logging systems allowing researchers to understand navigation patterns between web-pages. However, these studies have often required the participants to abandon their preferred web-browser and instead use a customised or open-source alternative that is equipped for logging. Asking participants to use an alternate browser to that which they are familiar raises issues of data validity. Users may interact in a different manner with a surrogate application that supports a subset of the real system features or which presents them in a different manner. The literature reports four main methods for logging activities in web-browsers: customising an existing web browser for logging, creating a custom web-browser, developing an external application that monitors the activities taking place within the web browser or adding a proxy that modifies and/or logs incoming web pages.

Catledge and Pitkow [39], and Tauscher and Greenberg [213, 214] both directly modified the source code of the XMosaic [175] web-browser, adding logging functionality and then distributing the new browser. Hawkey and Inkpen [98]

(and later Kellar et al. [121]) were able to extract “limited navigation events” from Microsoft’s Internet Explorer (IE) [156] browser by adding a *Browser Helper Object* (BHO) that was loaded every time the browser was started. This required the BHO, in the form of a Windows DLL, to be distributed and installed by participants. The need for more detailed logging abilities required researchers to develop their own custom web-browsers. Claypool et al. [48] created *The Curious Browser* to aid their research into implicit interest indicators. Kellar et al. [120, 121] wrote their own custom web-browser to characterise the factors that influenced web-browser navigation. Their browser mimicked the behaviour of Internet Explorer.

External loggers allow participants to continue using their everyday browser, but still have their actions logged. Turnbull [221] used the stand-alone tool *WebTracker* for monitoring web usage in Netscape 4.6 and IE 5.0. Reeder et al. [189] created *WebLogger* to log events in IE, however it was designed for the laboratory environment, with an IE window appearing as WebLogger was started. Ellis et al. [68] created *Listener*, a tool to log web navigation behaviour in Netscape Navigator on MacOS 7.5.1, using Apple Events.

A proxy server can log web-use without requiring installation of additional software onto a participant’s computer. Hong et al. [102] created *WebQuilt*, a proxy server for all incoming and outgoing web traffic. Atterer et al. [19] also placed a proxy between the client and the outside Internet. The proxy recorded all web transactions and added custom Javascript code to each web-page to enable client-side logging of user actions. They could record page views, the length of time spent on a page, mouse and keyboard actions, scrolling, and window-resizing. Atterer and Schmidt [18] extended this setup to work with AJAX applications. Weinreich et al. [225] used a client side intermediary that filtered all web pages that were directed to a web browser and inserted Javascript into the page to allow detailed reporting of links selected, form submission, browser history and page properties (however, this *did* require extra software installation).

Commercially available logging tools offer another option for researchers recording web-browsing behaviour. Kelly and Belkin [123] recorded all interaction with applications, including keystrokes, using the *WinWhatWhere* recorder (no longer available) for their study of web browser interactions. Kim and Allen [125] used Cistron’s *LittleBrother* [47] to record network traffic to web browsers.

Loggers employed to record user actions in Microsoft Word and Adobe Reader are of particular interest to this thesis. Microsoft Word, along with many Microsoft products, ships as part of the Microsoft Customer Experience Improvement Program (CEIP) [157]. The CEIP logs user actions and program state to aid application developers when designing improvements to the product. The CEIP works as an opt-in logging program to record command use, program performance and problems encountered. The Adobe Product Improvement Program [6] works in a similar manner. The majority of the data collected from these programs remains unpublished.

Creation of logging systems such as the CEIP requires access to the source code of the application. This approach works well if it is the product developer who wishes to log the application (this approach is often used by researchers creating new methods of interaction). However, it is impractical when a researcher wishes to log an application they did not create and for which they do not have the source code. McGrenere [149] describes two tools used internally by Microsoft: MTracker and IV, an Instrumented Version of Microsoft Office [153]. MTracker utilises the Microsoft Active Accessibility Interface to listen to events that are produced from many Windows based applications. IV uses hooks embedded in the source code of the Microsoft Office [153] application to log command use.

2.4 Summary and Discussion

The literature reports five broad user tasks that require document navigation: ‘over-viewing and browsing’, ‘reading’, ‘annotation and writing’, ‘searching’ and ‘re-visitation’. Some of these tasks are better supported by paper documents, others by their electronic counterparts. The physical nature of paper readily supports over-viewing and browsing tasks. Books and magazines can easily have their pages flicked; unbound documents can be spread on a table. Annotation is more natural on paper-based documents, as they provide a clear distinction between the original content and the additional markings. Searching and goto tasks are generally better supported by electronic documents—the work involved in finding a specific phrase or position within the document is offloaded to the computer.

These five navigation task categories are not fully complimented with the low-

level actions required to achieve the specified goals. For example, linear, receptive reading may require slow continuous scrolling in an electronic document or page turning in a paperback book. The primary reason for not reporting such actions is the absence of literature that breaks document navigation tasks into their constituent components—personal experiences can speculate on the actions involved, but little evidence is available to back this up. A characterisation of this type, while extremely useful for document navigation system designers, could prove impractical to create. For example, this chapter reports thirteen reasons for reading to take place, each of which may require different navigation actions. The diversity of users also means a simple, linear reading task would be performed in numerous different manners.

To understand document navigation, researchers have employed a variety of tools and techniques to observe and record user actions. These fall into two main categories: human-centric techniques and automated logging software. The biggest advantage of human-centric observations is the flexibility they allow. Researchers can probe comments of interest deeply to elicit informative user thoughts. Unfortunately, they are time-intensive, making long term studies impractical. Automated logging tools require less researcher input, but often lack context. One option for overcoming these issues is to employ a combination of both techniques. Researchers should use the empirical data recorded by loggers as input into interviews and surveys to examine areas that were revealed as interesting or unusual.

Many of the studies reported in this chapter detail the use of documents and document navigation in small, restricted observations. However, there are few, if any, that report document interaction over long periods of time. Studies that investigate how computer users divide their time among these activities and which are poorly supported would give interaction designers important insights into areas for navigation improvement.

Chapter 3

A Review of Electronic Document Navigation Tools

ELECTRONIC document navigation tools modify the view onto a document by adjusting the on-screen position, zoom, layout and orientation. Facilities for this kind of manipulation have existed since the first Graphic User Interfaces (GUIs), with Sutherland's *Sketchpad* [212] using dials to move the x- and y- coordinates and adjust the magnification of a drawing. The scrollbar, with origins in the *Smalltalk* and *Interlisp* environments [64], is the current de facto standard for electronic document navigation.

Proposed improvements to document navigation tools are reviewed in this chapter. The tools are presented in the eight categories that are summarised in Table 3.1; this table also illustrates their support in twelve common document navigation applications. The review proceeds as follows. It begins with a description of the core document navigation tools that are extensively deployed on desktop computers and which are precursors to many of the proposed improvements. Second, devices used for providing input for document navigation are introduced. Augmentations of the standard scrollbar—both visual and behavioural—are then presented, followed by tools that enhance navigation through awareness of the document's content. Document visualisations that present multiple views, techniques for indirect manipulation, zoom tools and revisitation tools complete the review. Next, document navigation on mobile-devices is briefly summarised, to expose techniques that could potentially inform the design of desktop-based navigation tools. An analysis of the procedures and frameworks for evaluating document navigation tools concludes the chapter.

		Application												
		Adobe Reader 7	Enacs/XEmacs 21.x	Evince Document Viewer 0.6	Internet Explorer 7	Microsoft Powerpoint 2003	Microsoft Word 2003	Mozilla Firefox 2.0	OO Impress 2.0.4	OO Writer 2.0.4	Vi/Vim 7.0	Visual Studio 2005	Xpdf 3.01	
Core navigation tools († = only in full-screen mode, ‡ = using print preview)														
1	Standard scrollbar	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	
2	The mousewheel	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	
3	Paging buttons	✓		✓		✓	✓	✓	✓	✓		✓	✓	
4	Paging keys	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	
5	RBS				✓		✓	✓		✓		✓		
6	Phrase search	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
7	Goto page/line	✓	✓			†	✓		✓	✓	✓	✓	✓	
8	Manual zoom tools	✓			✓	✓	✓	✓	✓	✓			✓	
9	Panning	✓		✓									✓	
10	Multiple document layouts	✓		✓	‡	✓	✓	‡	✓	✓				
Input devices (§ = if drivers available, ✕ = by manipulating interface artifacts, ▲ = unknown)														
11	TrackBall	§	§	§	§	§	§	§	§	§	§	§	§	
12	ScrollPoint	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
13	Pressure-sensitive mouse	▲	▲	▲	✓	▲	▲	▲	▲	▲	▲	▲	▲	
14	Arrow key navigation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
15	Stylus-based input	✕	✕	✕	✕	✕	✕	✕	✕	✕		✕	✕	
16	SlideBar	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	▲	
Scrollbar augmentations														
17	TES	✓		✓		✓	✓		✓					
18	Single thumbnail TES	✓												
19	Edit wear and read wear													
20	Thumbar													
21	Mural bars													
22	Code thumbnails													
23	Value bars													
24	Confetti scrollbar													
25	Auditory-enhanced scrollbar													
26	AlphaSlider													
27	FineSlider													
Content-aware navigation aids (♥ = the ability to follow internal links, ♦ = through functions such as go to definition, ▲ = bookmarks persist in separate files α = list matching lines, β = reviewing tools indicate changes, γ = serially, by using note and comment tools, δ = supported when using some navigation tools)														
28	Content-aware scrolling													
29	Intra-document links ♥	✓		✓	✓	✓	✓	✓	✓	✓		♦	✓	
30	Pre-defined bookmarks	✓	▲	✓			✓		✓	✓	▲	✓	✓	
31	First and last page	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
32	10 pages-at-a-time navigation												✓	
33	Semantic browsing						✓			✓		♦		
34	Context-based search	✓	α	✓			✓	✓		✓		✓		
35	Context lens													
36	Enhanced thumbnails													
37	Popout prism													
38	Diff tools		✓			β	β			✓	✓			
39	Reviewing tools					✓	✓		✓	✓				
40	Anchored conversations					γ	γ		γ	γ				
41	TVC													
42	GazeMarker													
43	Animated scrolling	δ	δ	δ	δ	δ	δ	δ	δ	δ	δ	δ	δ	
Document visualisations that provide multiple views (ε = Two views of the document in separate windows (content changes are synchronised), ζ = Not in version 7—this is provided in version 9, η = in two separate views)														
44	Split-views		✓				✓		ε	ε	✓	✓		
45	Separated overview+detail	ζ				ζ			η					
46	Thumbnail slider													
47	Z-lenses													
48	DeepDocument													
49	Fisheye view													
50	The document lens													
51	Flip-zooming													
Navigation using indirect manipulation (θ = continued depression in the scrollbar trough, § = if drivers available)														
52	SDAZ													
53	DDAZ													
54	RSVP	θ	θ	θ		θ	θ			θ		θ	θ	
55	Flipper													
56	MRSVP													
57	Hardware scroll ring	§	§	§	§	§	§	§	§	§	§	§	§	
58	Virtual scroll ring													
59	Radial scroll tool													
60	Curve dial													
61	Two-handed navigation													
62	Perspective view navigation													
63	Navigation using eye-tracking													
Zoom tools														
64	Semantic zooming													
65	Zliding													
66	OrthoZoom scroller													
67	Space-filling thumbnails													
Revisitation tools (λ = between pages)														
68	User defined bookmarks		✓		λ		✓	λ		✓	✓	✓		
69	Bookmark scrollbar													
70	Thumbnail trays													
71	Electronic dog-ears													
72	Click-through navigation													
73	Previous view and next view	✓			λ			λ				✓	✓	

Table 3.1: Application support of document navigation tools. Full details of the applications surveyed are available in Appendix B.

3.1 Core Navigation Tools

A small set of tools make up the core navigation aids found in the majority of today’s document navigation applications. These tools are presented together in this section to lay the foundations for many of the suggested improvements.

3.1.1 The Scrollbar

The *scrollbar* is an interface widget that displays and controls position within a document. Virtually all present-day computer users are familiar with the scrollbar, due to its widespread adoption, not only for document navigation systems (Table 3.1, line 1), but also for navigating virtually any data-set that is too big to fit on-screen. For completeness and to detail naming conventions, the scrollbar’s operation is briefly described below.

A “standard” scrollbar consists of four primary components, as illustrated in Figure 3.1. The *trough*¹ extends the length of the widget, with an *up arrow* and a *down arrow* at either end. The size of the trough maps to the length of the document—the highest point to the start of the document; the lowest point to the end of the document. The *thumb*² is a rectangular region that moves within the trough. It visually conveys two pieces of information about the document. First, the extent of the thumb is proportional to the current segment of the document that is visible on-screen. Its size will be influenced by the document length and current zoom, although a minimum size is maintained for ease of acquisition. Second, it displays the current position within the document.

The standard scrollbar supports a number of interactions for modifying the current document position: dragging the thumb to another location, clicking in the trough to invoke paging and pressing the arrow buttons for fine-grained movement. Dragging the thumb will usually (but not always) cause the window’s content to update, in real time. Not doing so will be “disconcerting and negatively impact a users ability to navigate content inside the view” [218, pg. 90]. Clicking in the trough will cause the content to be updated a *screenful* at a time. The definition of screenful is application dependent, but often is a complete replacement

¹Also known as the *track* [170], *scroll track* [16] or *shaft* [167].

²Also known as the *scroller* [16], *knob* [211] or *scroll thumb* and sometimes informally as the *bar*, *scroll box* or *elevator*.

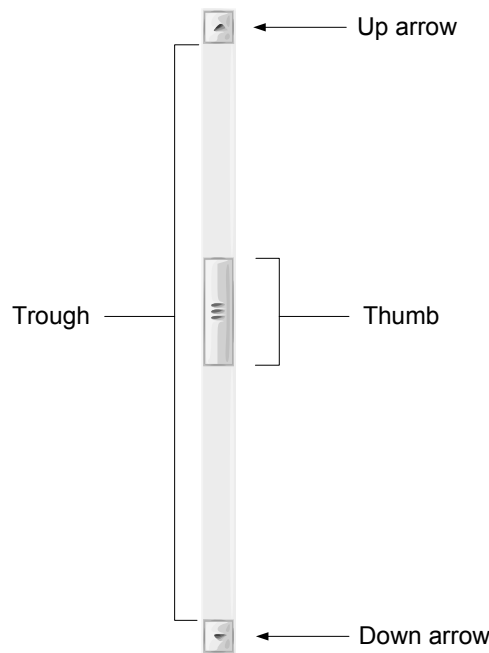


Figure 3.1: Standard vertical scrollbar components

of the current contents with those directly above or below. In the context of document navigation, it may also mean a page. Pressing and not releasing the mouse button in the trough will cause the thumb to move to that position. The arrows (also known as *Repeat Buttons* [170] or *Scroll Arrows* [16]) move the thumb in the direction of the arrow a *line* at a time (in text-based documents a line is exactly that, for other content it may differ).

3.1.2 The Mousewheel

The *mousewheel* is a common addition to standard mice that is widely supported for scrolling activities (Table 3.1, line 2). The addition of a wheel to the mouse was first described in 1993 by Venolia [224] to aid the task of moving in 3D space. Their *roller mouse* had a single button and two wheels connected by an axle. Wheel rotation controlled cursor movement in the Z-plane, moving it closer to, or further from the screen. Commercial development of the mouse resulted in a single wheel (that often also acts as a button) positioned between the left and right buttons. It is intended for use with the middle finger on a human hand. Other

designs have introduced wheels on the side of a mouse for use with the thumb and tilt-wheels that allow horizontal scrolling.

Employing the mousewheel when reading a document means users do not need to acquire the scrollbar, an action which distracts readers, potentially causing them to lose their position. Typically, the wheel has a series of notches which are felt during rotation. A single “click” of the mouse-wheel will move a document vertically by a preset amount (typically around three lines per notch [100]).

Overuse of the mousewheel to move long distances prompted work into wheel acceleration and momentum. Hinckley et al.’s [100] study of scrolling techniques compared an isometric joystick, standard mousewheel and two mousewheels with acceleration applied. The authors found that “the performance of the wheel can be significantly improved using an acceleration algorithm” [100, pg. 65]. Recent commercial advances have moved the mousewheel from a small distance movement device into one that can also comfortably move long distances. For example, Logitech’s hyper-fast mousewheel [136] has two modes—a standard ratchet mode and a momentum mode. In momentum mode, the heavier than normal mouse wheel spins freely, moving through many pages at a rapid rate.

Recognising the value of the mousewheel, laptop computer manufacturers often include a mousewheel emulator on touch-pad devices. To activate this feature, users move their finger over dedicated regions of the touch-pad (usually the bottom each for horizontal scrolling and the right-hand edge for vertical scrolling) or perform a gesture intercepted by the operating system (such as the two-finger scrolling action on some Apple laptop computers [15]).

The term *scrollwheel* has arisen from the most commonly performed task with the wheel—scrolling. The terms mousewheel and scrollwheel are used interchangeably. This thesis uses the term *mousewheel* throughout—instances where it is used for tasks other than scrolling will be described later in this chapter.

3.1.3 *Paging*

Paging, analogous to turning the page of a book, allows the user to move through the document one page at a time. Traditionally, this is performed using the page up and page down keys on the keyboard. Continuous depression of these keys triggers an auto-repeat function that allows very fast movement through a docu-

ment, creating an RSVP-like effect (see Section 3.6.1). Applications may interpret *paging* as an action to move to the start of each consecutive page, regardless of the percent shown on screen, or to replace all of the contents on-screen with the screenful that precedes or follows. Interface designs may also display *paging buttons* on the screen to allow this same action to be performed using the mouse. Both of these mechanisms for paging are well supported in current navigation systems (Table 3.1, lines 3–4).

Page turning methods employed in *Realistic Books* emulate the act of turning a physical page in book, helping to bridge the gap between paper and electronic documents. Instead of pressing a key or interface button, the mouse is used to drag the corner of a page in the same manner that the corner of a physical book would be grabbed and turned using the thumb and finger. Liesaputra and Witten [133] studied information-seeking tasks in a Realistic Book, HTML and PDF document formats. For these tasks, The Realistic Book was significantly faster and subjectively preferred over the other two techniques.

3.1.4 *Rate-based Scrolling (RBS)*

Rate-control techniques provide rate or velocity as input, instead of position. The most common example of *rate-based scrolling* is the “middle click and drag” gesture available in some Microsoft Windows applications, most notably the Microsoft Office Suite [153] (Table 3.1, line 5). Users have two gestures available for triggering this navigation tool: click the middle mouse button and then drag or press, drag and release. The rate-based scrolling icon appears on gesture initiation (Figure 3.2). The distance the mouse moves away from the initiating position determines the rate of movement—the further from the starting position, the faster the movement. The scrolling action is terminated by returning the mouse to the initial position, clicking the middle button (if using the first method) or simply releasing the button at the appropriate time.

3.1.5 *Search Tools*

Locating a specific piece of information in a document is often tedious. Search tools allow the user to find occurrences of a particular object within a document. The most common form of search is *Phrase Search*, (also commonly referred to



Figure 3.2: Rate-based scrolling icon

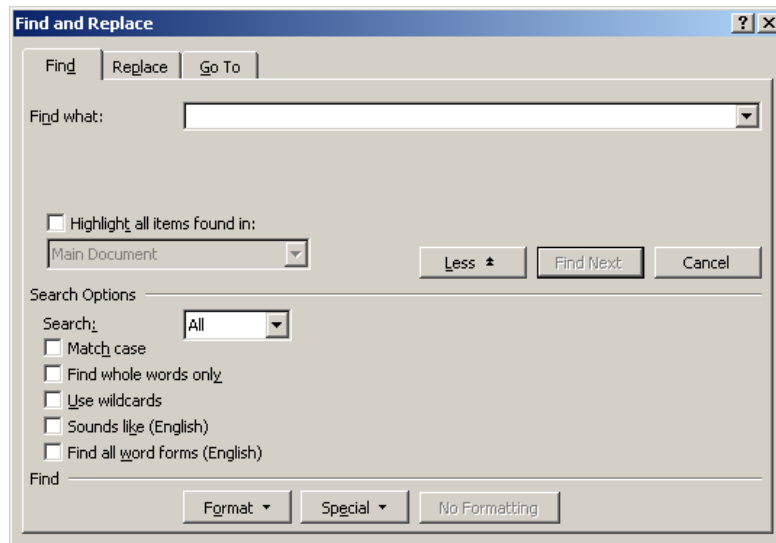


Figure 3.3: Microsoft Word's *Find and Replace* dialog box

as the **Ctrl-f** function [137]) that is supported by all surveyed document navigation systems (Table 3.1, line 6). The user enters part or all of the word or phrase required, the application searches the document and then lists or cycles through all of the occurrences. Most applications provide further filtering options, including case matching, full word matching and search direction (backwards or forwards from the current position). The use of regular expressions is not widely supported. Figure 3.3 shows the *Find and Replace* dialog box from Microsoft Word. It provides a text entry form for the search phrase, a choice of where to search and a list of search options. The search options include the ability to locate the phrase in text which is formatted in a particular manner, for example, searching only within text formatted in the *heading* style.

Text-based search features are considered to be one of the main advantages of electronic content over paper based documents. Loizides and Buchanan [137]

investigated this belief through three user studies. The majority of participants agreed the Ctrl-f feature was important in electronic searching; however, in practise they found that the feature was not used as often as expected. Analysis of user behaviour revealed the problem of “not knowing what to look for”, spelling mistakes causing direct matches to fail and the ease of using web-based search engines for *within* document search (search term precision is less essential).

Advanced search tools also allow searching and browsing between non-text document objects. This is discussed in Section 3.4.3.

3.1.6 *Goto Tools*

Goto tools allow the user to exploit their pre-acquired knowledge of the document to move directly to a particular, well-defined position. The most common example of goto functionality is the *Goto Page* tools supplied with most document navigation systems (Table 3.1, line 7). The user enters their desired page number and the system takes them directly to that page. This function is either implemented as a keyboard shortcut (such as Ctrl-g in Microsoft Word), or via an editable page indicator (such as in Adobe Reader). Some systems also provide *relative goto* page functionality—the user can enter a positive or negative number to move either forward or backward by a certain number of pages.

In a similar manner to search tools, advanced goto tools allow non-text objects to be quickly acquired (for example, “goto the second table in the document”). Section 3.4.3 discusses these.

3.1.7 *Zoom Tools*

A document’s zoom determines how close the document appears to the screen, which influences the proportion of content that is visible. Zooming is likened to adjusting the distance between the document and a camera that sits perpendicularly above it [92]. The tools described in this section allow the user to manually modify the document zoom. These tools are supported in the majority of document navigation systems (Table 3.1, line 8).

Zoom focus	Zoom level selection	
	<i>Continuous</i>	<i>Discrete</i>
<i>User selected</i>	Dynamic zoom Scrollbar zoom Overview zoom	Point-and-click zooming Drag-to-zoom
<i>Application selected</i>	Ctrl-Mousewheel Sliders	Drop down menus Keyboard shortcuts Menu selections Increase/decrease buttons

Table 3.2: Manual zoom tool classification

Properties of Zoom Tools

There are two important properties of manual zoom tools: *zoom level selection* and *zoom focus selection*.

Zoom level selection refers to the nature of adjustability of the tool—either continuous or discrete. Continuous zoom tools allow a smooth transition between a series of zoom levels, such as a slider or the Ctrl-Mousewheel technique. Discrete zoom tools adjust the zoom to a discrete pre-defined level, such as from a drop down menu, or *click to zoom* button. Continuous zoom adjustment methods usually animate the display through the zoom levels as adjustment is occurring, allowing users to more easily orient themselves in the new view. Discrete methods move immediately to the requested zoom level.

The zoom focus defines the point in the document around which the zoom action occurs. This is either user selected or pre-determined by the application. Users may identify this position by selecting a center point or defining the edges of the zoom action. Applications may choose to use the center of the current contents, the cursor position or by keeping the left, right, top or bottom edge constant. Many interaction techniques in commercial, open-source and research systems exist for the task of manual zooming. These are classified in Table 3.2 according to the two properties identified here.

Continuous, User Selected Focus

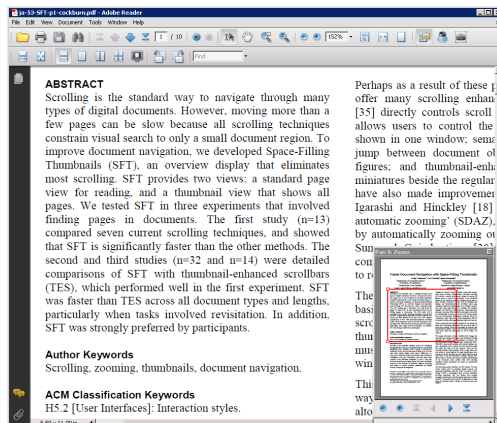
Dynamic Zoom is a technique introduced in Adobe Reader that allows the user to continuously change the document zoom after selecting the focal point. Once in dynamic zoom mode, the user selects the point of interest and drags the mouse cursor vertically. Movement towards the top of the screen zooms in, towards the bottom zooms out. *Scrollbar Zoom* systems allow the zoom of the document to be altered by manually adjusting the length of the scrollbar thumb. In the year 2000, Laakso et al. [129] suggested that the length of the scroll thumb be modified by dragging the borders, altering the level of detail on an electronic calendar. In 2007, Song et al. [205] presented the *Active Scrollbar* that allowed the user to drag arrows positioned on either end of the thumb to alter a document's zoom. *Overview zoom* allows the user to modify the zoom of the main document pane by adjusting the size of a view in an auxiliary overview panel (see Figure 3.4a).

Continuous, Application Selected Focus

The *Ctrl-Mousewheel* technique and *zoom sliders* allow continuous zoom adjustment, but have points of focus selected by the application. Ctrl-Mousewheel zooming requires the user to hold down the Ctrl key and then scroll the wheel on their mouse. The relationship between the direction of wheel movement and the increase or reduction of zoom is application dependent. The focus for this tool is the cursor position, the mouse pointer position or the center of the current document view. When it does not conflict with other functionality, the mousewheel is used by itself to control zooming (such as in the Google Maps [81] application). Zoom sliders allow the user to modify the document magnification by dragging a slider (for example, Figure 3.4b). The mouse pointer is occupied over the slider during adjustment, meaning any user choice of zoom position is difficult (unless it is pre-selected).

Discrete, User Selected Focus

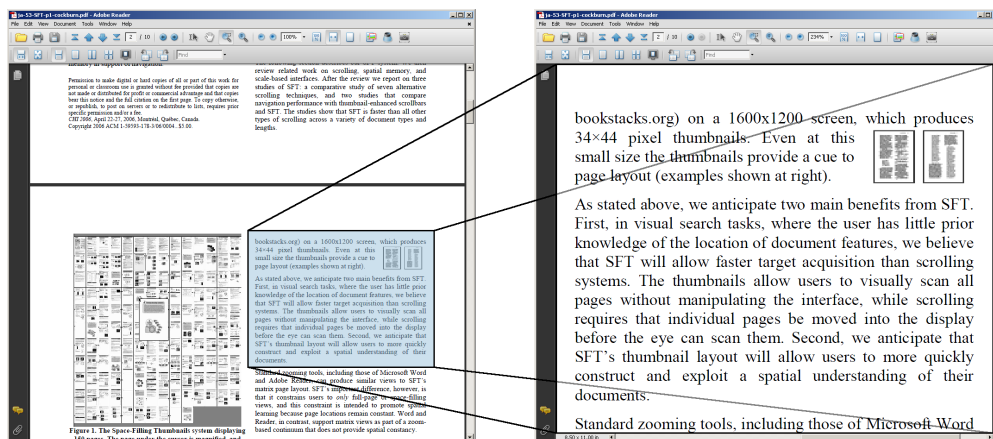
A *point-and-click* zoom mode allows the user to select a point of interest on the document and click or double-click to zoom in or out by a preset amount. *Drag-to-zoom* allows the user to select an interesting region for zooming by dragging over an area (Figure 3.4c, left). The application then adjusts the viewport so only



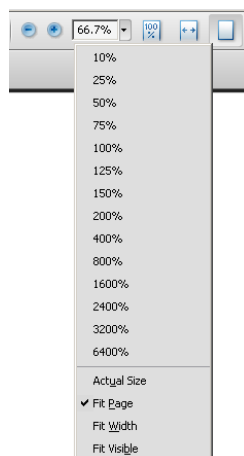
(a) Adobe Reader's overview zoom panel (bottom right). The page's zoom and position are changed by resizing and dragging the red box.



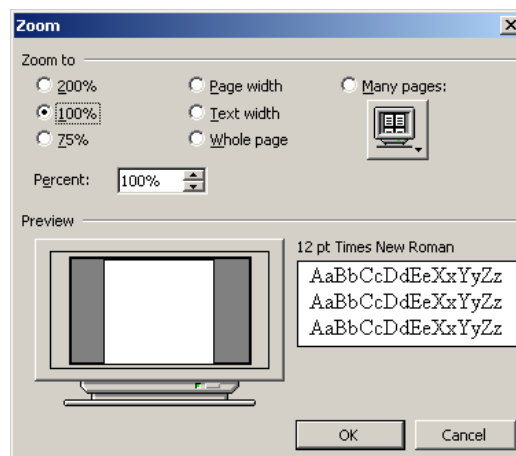
(b) Microsoft Word's zoom slider



(c) Adobe Reader's marquee zoom. The region of interest is first selected (left) and is then zoomed to fit the screen (right).



(d) Adobe Reader's zoom drop-down menu



(e) Microsoft Word's zoom adjustment dialog box

Figure 3.4: Zoom controls

that region is visible on screen, and increases the zoom appropriately (Figure 3.4c, right). This is analogous to cropping an image to remove unwanted content (although in this case the content is hidden, not removed).

Discrete, Application Selected Focus

Finally, there are several methods of discrete, application selected point-of-focus zooming. Drop-down menus (Figure 3.4d) allow the user to select a zoom percentage (for example, 150%) or a zoom dependent on the window size on screen (for example, *fit page width*). Menu selections allow the same options, usually also with the ability to enter a freeform zoom value (Figure 3.4e). Keyboard shortcuts and increase/decrease buttons also modify the zoom level by a fixed amount.

3.1.8 Panning

Panning tools allow vertical movement and/or horizontal position adjustment via direct manipulation. For example, Adobe Reader users can enter pan mode, press, drag the document in any direction and release the left mouse button. There is a 1:1 mapping between the pixels moved by the mouse and those of the document on-screen, limiting the single action distance. Panning is often used in conjunction with zoom tools—panning in a zoomed out view moves across the information space much faster than when zoomed in. Panning tools are implemented in all of the surveyed PDF readers, but not in applications that support editing (Table 3.1, line 9).

3.1.9 Document Layout

Layout tools modify the presentation of a document's content on-screen. These tools manipulate display properties such as the number, size, rotation and zoom of a document's view, the display of *non-printable* characters (such as spaces, line breaks and page breaks) and visualisations of the document on a different medium. Many document navigation systems support multiple document layouts (Table 3.1, line 10). These are briefly summarised below.

A *Book View* aligns two sequential pages of a document beside each other, as if the reader was looking at the pages of an open book (see Figure 3.5a). This is an advantage on high resolution widescreen monitors, as it can utilise otherwise

unused space, while maintaining a familiar layout. A *Print View* a rendering of how a document will look when printed. This will take into account factors such as paper size, colour/grey-scale settings and margins. *Web Views* illustrate the document's appearance when displayed in a web browser. An *Outline View* is appropriate for writing tasks— it visually displays non-printable characters, such as spaces, paragraph, line and page breaks (see Figure 3.5b).

Rotation tools generally allow the user to modify the presentation angle by 90° at a time, facilitating changes from portrait to landscape view.

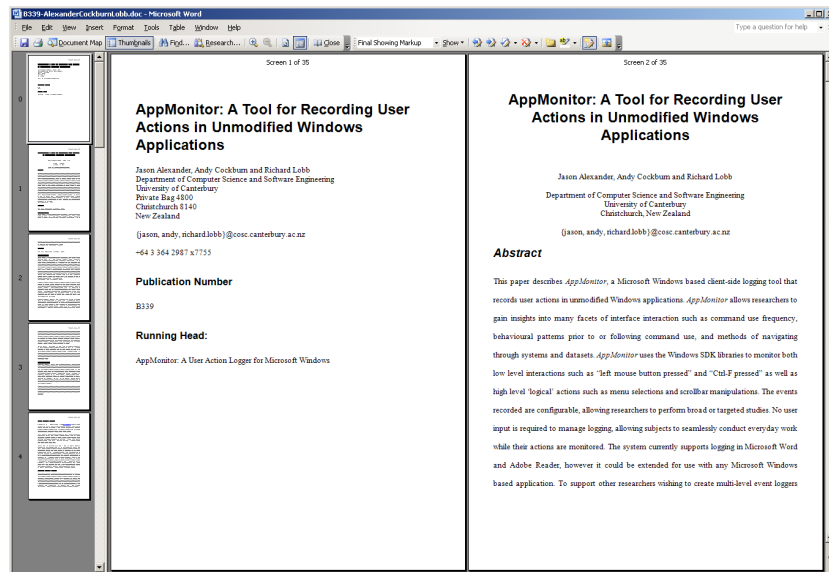
3.2 Input Devices

Humans communicate with computers using input devices; typically a mouse and keyboard. The mouse was invented by Engelbart [69] in 1967 as a method for on-screen text manipulation. Since this time, it has evolved into the present day incarnation (Figure 3.6b), with wheels, buttons, ergonomic design and optical tracking. The keyboard has a longer history. The first patent was issued in 1714 to Mill who described a machine for “impressing or transcribing of letters” [24, pg. 4]; it was not until 1867 Christopher Sholes received a patent that led to the first commercial typewriter. Present-day keyboards, though mainly electronic instead of mechanical, provide the same interaction style: keys are pressed to activate an input (Figure 3.6a). However, many additional keys that were not available on typewriters are now provided, for example *page up/page down* and the arrow keys. This section describes how the mouse and keyboard, along with newly designed input devices, are used for document navigation tasks.

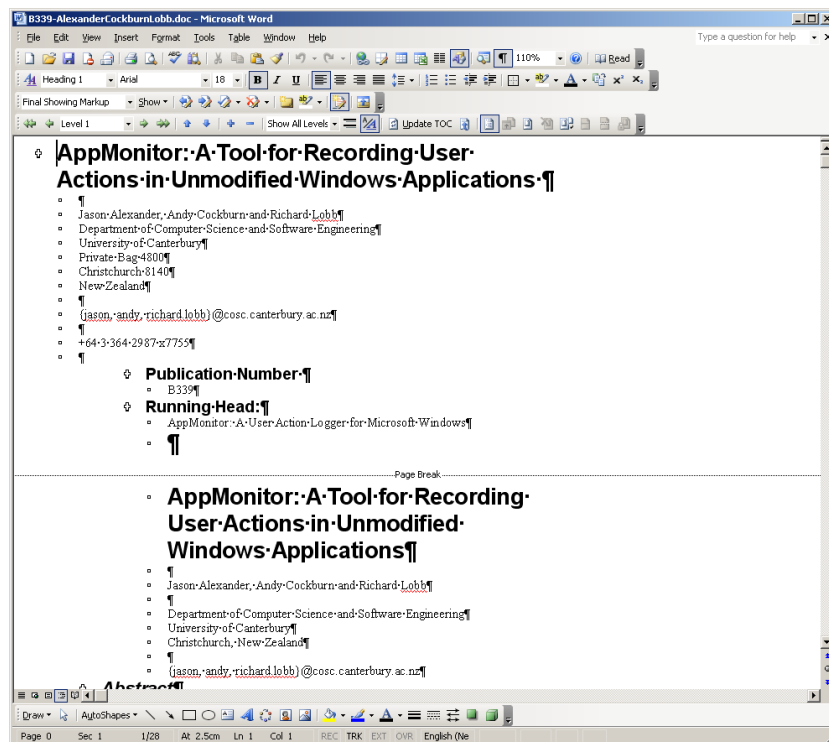
3.2.1 Mouse Activated Document Navigation

The mouse provides facilities for on-screen pointing and target selection. It allows precise positioning of a cursor in 2D space, with selections made using one or more buttons placed on the surface of the mouse.

Many on-screen document navigation tools rely on direct manipulation [107] with the mouse to facilitate view modification. These actions are divided into three stages: acquiring the on-screen target, performing an action with the target (dragging, selecting or clicking) and if applicable, releasing the target.



(a) Book view



(b) Outline view

Figure 3.5: Two of Microsoft Word's document views



Figure 3.6: Common desktop input devices

The target acquisition stage is well described by Fitts' law [74] and is widely verified (for example, Card et al. [38] and MacKenzie et al. [142]). In essence, Fitts' law states that objects that are smaller or further away will take longer to acquire than those that are larger or closer. Equation 3.1 shows MacKenzie's [141] widely accepted improvement to Fitts' law. The movement time, MT , is proportional to the *Index of Difficulty*, ID . ID is determined from A , the amplitude of movement and W , the target width. The constants a and b are determined by linear regression.

$$MT = a + b \times ID \quad (3.1)$$

where:

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

All interface widgets that require acquisition with a mouse are subject to this movement time. Small widgets placed a long distance from the document focus may unnecessarily penalise navigation time. Several researchers have modeled target acquisition with document navigation tools, such as the scrollbar, using Fitts' law. This is discussed further in Section 3.10.4.

3.2.2 Mouse Appendages and Augmentations

The mousewheel is the most commonly observed addition to the standard point and click mechanisms on a mouse (see Section 3.1.2). However, input device vendors have developed several other appendages, to aid scrolling tasks.

A *TrackBall* is a small ball embedded into a mouse, allowing the user to scroll in any dimension, rather than the single (vertical) dimension supported by the standard mousewheel. If required, a trackball can then allow the shell of the mouse to remain stationary, with the ball used for all navigation. Typical examples of mice that include trackballs are Apple's *Mighty Mouse* [12] and the Logitech *TrackMan* [135].

The ScrollPoint mouse [108] is a commercial version of the mouse prototyped by Zhai et al. [236]. A miniature, pressure-sensitive isometric joystick is mounted between the left and right mouse buttons. The joystick is a rate-control device that controls the *velocity* of scrolling, rather than the position. The harder the joystick is pressed, the greater the velocity input. Zhai et al.'s study found that ScrollPoint was faster than the scrollbar and the mousewheel for visual search tasks [236]. ScrollPoint is supported by any application that supports the mousewheel (Table 3.1, line 12).

Pressure-sensitive mice allow simultaneous adjustment of a parameter and selection with the cursor. Cechanowicz et al. [41] augmented and evaluated a mouse with one and two pressure sensors. They found that the thumb and middle finger are the best points for input, while two sensors can provide greater control than one. The *Inflatable Mouse* [126] is housed in a laptop computer's card slot and inflates when required for use. The user can deform the balloon as a method of providing pressure sensitive input to the computer. Further, touch sensors at locations on the balloon can sense the direction from which the pressure is applied. Kim et al. [126] suggest several navigation interactions: squeezing the sides could adjust the zoom level, the pressure of the squeeze could control rate-based scrolling velocity and applying pressure to one side of the mouse could engage horizontal scrolling.

3.2.3 Keyboard Based Navigation

Two sets of keys are commonly used for document navigation: the arrow keys for small distance movement and the paging keys for 'page at a time' navigation (see Section 3.1.3). These keys provide discrete input; auto-repeat functionality is engaged when a key is held depressed.

The arrow keys are used to move the document position at a slow linear speed

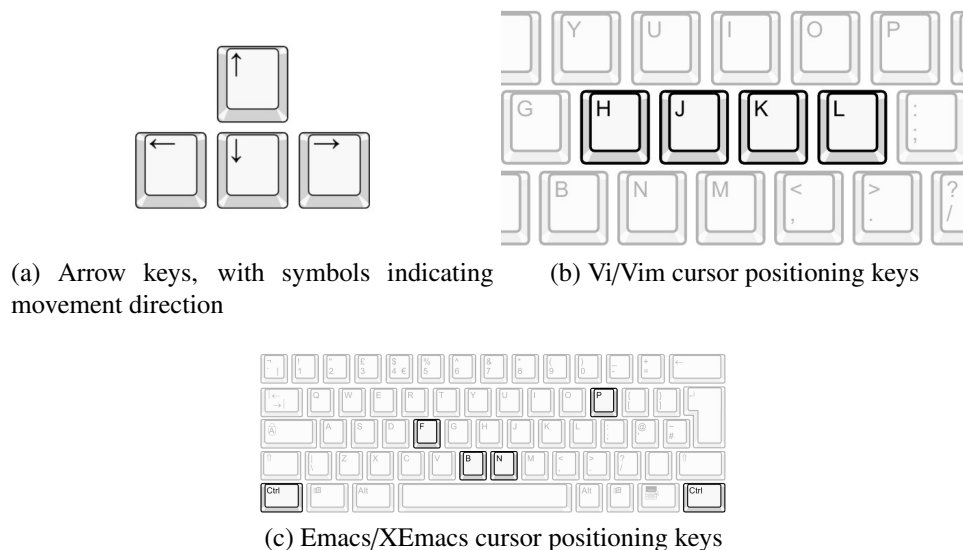


Figure 3.7: Keyboard-based navigation

(using auto-repeat) or to precisely position the cursor for editing. The four arrow keys on a keyboard provide 2D positioning (see Figure 3.7a). These keys move the cursor one *unit* at a time. A vertical unit is typically a line and a horizontal unit is typically a single character. Commonly, the `Ctrl` modifier key is applied to enlarge the size of the unit. Vertical units increase to the size of a paragraph and horizontal units increase to the size of a word. These distances are software dependent and may vary between applications.

Some programs also provide alternative keys for cursor positioning for efficiency or backwards compatibility. Along with the arrow keys, Emacs supports `Ctrl` and `f` (forward), `b` (backward), `n` (next line) or `p` (previous line) key-combinations for navigation (see Figure 3.7c). The GNU Emacs Manual [78] indicates that “it is faster to use these control keys than move your hand over to the arrow keys”. A similar idea applies in the Vi application: the `h`, (left) `j` (down), `k` (up) and `l` (right) keys are used instead of the arrows (see Figure 3.7b).

Users may also employ the cursor as an aid when reading—its position within the text providing a reference point if they are distracted and later return to the document.

3.2.4 Stylus-Based Input

A stylus is a pen-like device that is used on a special surface to convey both position and pressure as inputs. Traditionally, a stylus was used on a pressure sensitive tablet positioned on a user's desk. Advances in technology mean that laptop computers are now shipped with pressure sensitive screens, allowing 'drawing' to take place directly on top of the screen's content. The pressure input from a stylus can be used to adjust zoom level (see Section 3.7.2).

One gesture easily conveyed with a stylus, but not with a mouse is *flicking*. Flicking consists of three actions: pressing the stylus onto surface, dragging and releasing. Aliakseyeu et al. [8] describe four types of mappings from flick to document scroll speed:

Multi-Flick Add (MFA): The rate and direction of the flick increases and decreases the scrolling speed. Successive flicks add to or subtract from the current speed, with a tap stopping the movement.

Multi-Flick Standard (MFS): Directly maps the flick speed to the scroll speed.

Multi-Flick Friction (MFF): Is similar to MFS, except that it includes an additional friction element that gradually reduces the document scrolling speed after a period of time. This is the flick navigation mechanism employed on the iPhone [14].

Compound Multi-Flick (CMF): Provides visual feedback to the user during the drag and flick action (the other methods do not begin scrolling until the stylus is released). When the pen moves across the surface, the document scrolls a distance equivalent to that of the pen's movement. When the stylus is lifted, the flick speed maps to the scroll speed.

Aliakseyeu et al. found that CMF can perform as well as a scrollbar for short text documents, but is outperformed in longer documents.

3.2.5 SlideBar

The *SlideBar* [45] is a hardware slider that allows absolute position control within a document. The SlideBar knob has a movement range of 4.5 cm and is designed

to be placed beside a computer's keyboard. The knob position maps directly onto document position. The SlideBar performed better for visual search and Fitts' law tasks than a traditional scrollbar and was subjectively preferred over the scrollbar and mousewheel. The main disadvantage of such a device is the problem that occurs when the document position is modified by a method that does not involve the SlideBar (such as using the scrollbar). The document must move to the SlideBar's current position before further use—possibly leading to user confusion.

3.3 Scrollbar Augmentations

Scrollbar augmentations modify the appearance or behaviour of a standard scrollbar. Generally, these modifications take the form of additional panels, in-trough visualisations or changes to behaviour. This section documents each of these types of augmentation.

3.3.1 Thumbnail-Enhanced Scrollbars (TES)

Thumbnail-Enhanced Scrollbars (TES) augment a standard scrollbar viewport with a vertical column of thumbnails. All of the pages in the document are miniaturised and placed within a secondary scrolling environment. A common example of TES is in Adobe Reader, as shown in Figure 3.8, the auxiliary thumbnail pane is shown on the left-hand side of the window. Usually, the thumbnails follow any movement in the main window, however the main window only mirrors movement in the thumbnails when a particular page is selected. In their evaluation of seven document navigation systems, Cockburn et al. [53] found that TES of this type performed third fastest overall, and was faster for both visual search and spatial relocate tasks than traditional scrollbars.

Thumbnails may also appear dynamically when a scrolling action is underway. Adobe Reader provides this second type of TES when scrolling in *single page* mode (see thumbnail near the scrollbar in Figure 3.8). In this implementation, the primary view does not move when the main scrollbar is manipulated, but instead the thumbnail shows a miniature of the page that will be shown when the scrollbar widget is released. This snapshot was taken while a scrolling action is underway.

Thumbnail-Enhanced Scrollbars are supported by most applications that have logical page divisions (Table 3.1, lines 17–18). Single thumbnail TES (as in Fig-

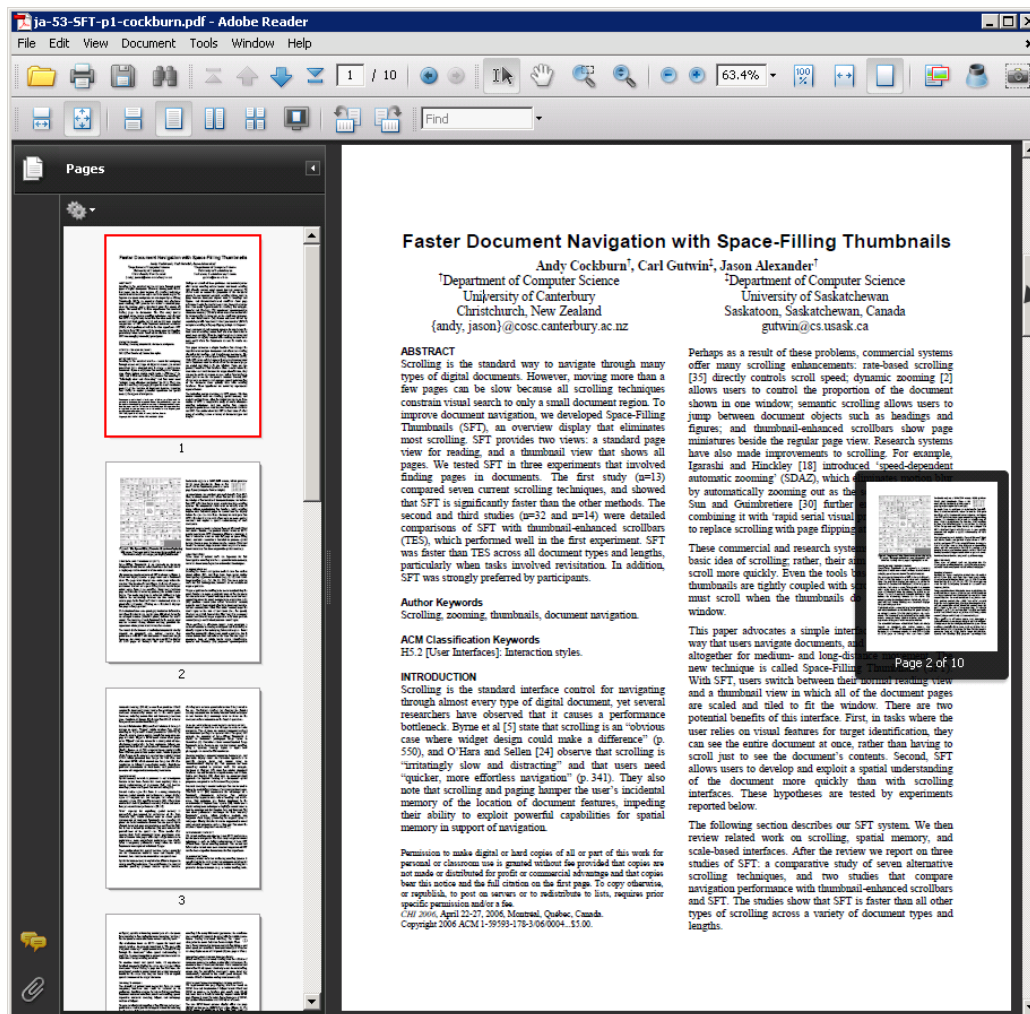


Figure 3.8: Adobe Reader’s thumbnail-enhanced scrollbars. The auxiliary thumb-nail panel is shown at left, the primary scrolling environment on the right. A small thumbnail also appears beside the scrollbar when scrolling in *single page mode*.

ure 3.8) is only supported by Adobe Reader.

3.3.2 Scrollbar-Based Visualisations

Visualisations within or beside the scrollbar allow users to see the standard scrolling widget adorned with useful information. The functionality of the scrollbar may remain identical or be modified to allow interaction with the visualisation.

Edit Wear and Read Wear

Hill et al.'s [99] *edit wear* and *read wear* allow users to quickly identify the amount of editing or reading that has occurred in any part of the document. A histogram-based visualisation is placed inside the scrollbar trough. It shows usage patterns within the document, allowing fast re-acquisition of positions of interest: those where heavy reading or editing occurred.

The scrollbar trough is divided into horizontal sections, based on the number of lines in the document. The area in the trough between the left and right edges of the scrollbar is used to indicate the amount of “wear” that has accrued. Read wear is determined by the length of time spent at each position within the document; edit wear by the number of edits made to each line. The scrollbar trough can be divided vertically when both types of wear are to be displayed simultaneously.

Information Space Visualisation

Information space visualisations allow the user to very quickly overview the document's content. Several researchers have augmented the scrollbar by placing a visualisation of the information space in a vertical column beside the main presentation pane.

Thumbar [86] was developed as part of the *Reader's Helper*. It places a scaled version of an HTML³ document in a column on the left-hand side of their web-browser. The user can drag a lens on the Thumbar to adjust the document position. The Thumbar allows readers to look ahead in the document and scan for pertinent regions. For long webpages, the Thumbar may require scrolling to view the complete visualisation.

³HyperText Markup Language

McCrickard and Catrambone [148] suggested *Mural Bars* as a method for providing an overview of the entire information space. In their implementation, the scrollbar is widened and a visualisation of the document (in their case source code) is placed into the trough. The thumb is modified to be entirely translucent, apart from the outline. In a similar vein, DeLine et al. [60] introduced *Code Thumbnails*, a technique that shrinks the complete contents of a source code file to fit onto the screen beside a standard scrollbar. The text in long files is unreadable; however, the authors suggest that users would build up their spatial memory of the workspace and very easily be able to move to the required position in the document by clicking inside the visualisation. When the mouse is hovered over the visualisation, the names of functions within the code are revealed at a readable zoom.

Identification of Salient Properties

Visualising salient properties provides the user with a method for judging the distribution of one or more attributes, helping to determine the document's relevance.

Large text-based listings are often difficult to scan for approximate attribute values. Chimera [44] introduced *Value Bars*, a tool that visualises attributes of a listing (for example, a UNIX directory listing) in a column beside the standard scrollbar. The weight of an attribute for a record is converted into a height in the value bar. Highly valued attributes correlate to large shaded regions in the value bar.

Researchers into information retrieval have considered in-trough visualisations for displaying their results. These visualisations originate from Wroblewski et al.'s [233] work on *attribute-enhanced scrollbars*. They proposed adding markings of different shapes, sizes, shades or colours to represent important documents segments. Byrd [36] describes one such system, the *Confetti Scrollbar*. Each occurrence of a particular term in the document is indicated by a coloured piece of "confetti" in the trough. The confetti's vertical position correlates to its location within the document (as well as being highlighted in the text). A mark is placed at a horizontal position within the trough to indicate approximate position on the line. A small evaluation of the system was performed, comparing systems with and without the trough visualisation. No significant differences were found, al-

though users preferred the system with the “confetti”. The Google Chrome [83] web-browser implements a simplified version of this system, by placing horizontal marks in the scrollbar at the position of search results (Figure 3.9a).

In a similar manner, software development environments such as Eclipse [217] and Netbeans [177] provide information regarding the state of the source code either in the scrollbar or in an additional trough beside the scrollbar. Examples of compile errors, search results and warnings from these two IDEs are shown in Figures 3.9b and 3.9c. Users are able to left-click on the marks to move directly to that position in the document—the auxiliary visualisation column allows additional behaviour to be added without the concern of modifying the scrollbar’s familiar behaviour.

Auditory-Enhanced Scrollbar

Brewster et al. [30] identified three problems that occur when navigating using the scrollbar: dragging the thumb outside its specified bounds (causing the document not to scroll), a lack of position awareness and “kangarooing” [30, pg. 175] when clicking in the scrollbar trough. First, if the user drags the thumb outside the scroll trough (either out the top, bottom, left or right), the document reverts to the original location, and is not scrolled. This behaviour is useful for quick revisitation (see Section 3.8), but is sometimes accidentally invoked. Second, in long documents, it is difficult to maintain a sound knowledge of your current position, especially when rapidly scrolling. The thumb provides an approximate representation of position, but requires specific user focus. Finally, kangarooing can occur when clicking in the scrollbar’s trough to move the thumb. In some situations the document will ‘snap’ to the contents, causing the thumb to jump between positions. If the user is not paying attention to the scrollbar, it may result in unknowingly jumping back-and-forward between two pages, wasting time.

The *Auditory-Enhanced Scrollbar* [30] addresses these issues by complimenting a standard scrollbar with audio output. Two tones are heard for every scroll action. A fixed tone is played when a document is scrolled; a low pitch when moving towards the end of the document, a high pitch when moving towards the start. The user will immediately be aware if kangarooing has begun. Status information regarding the current position in the document is provided by varying the

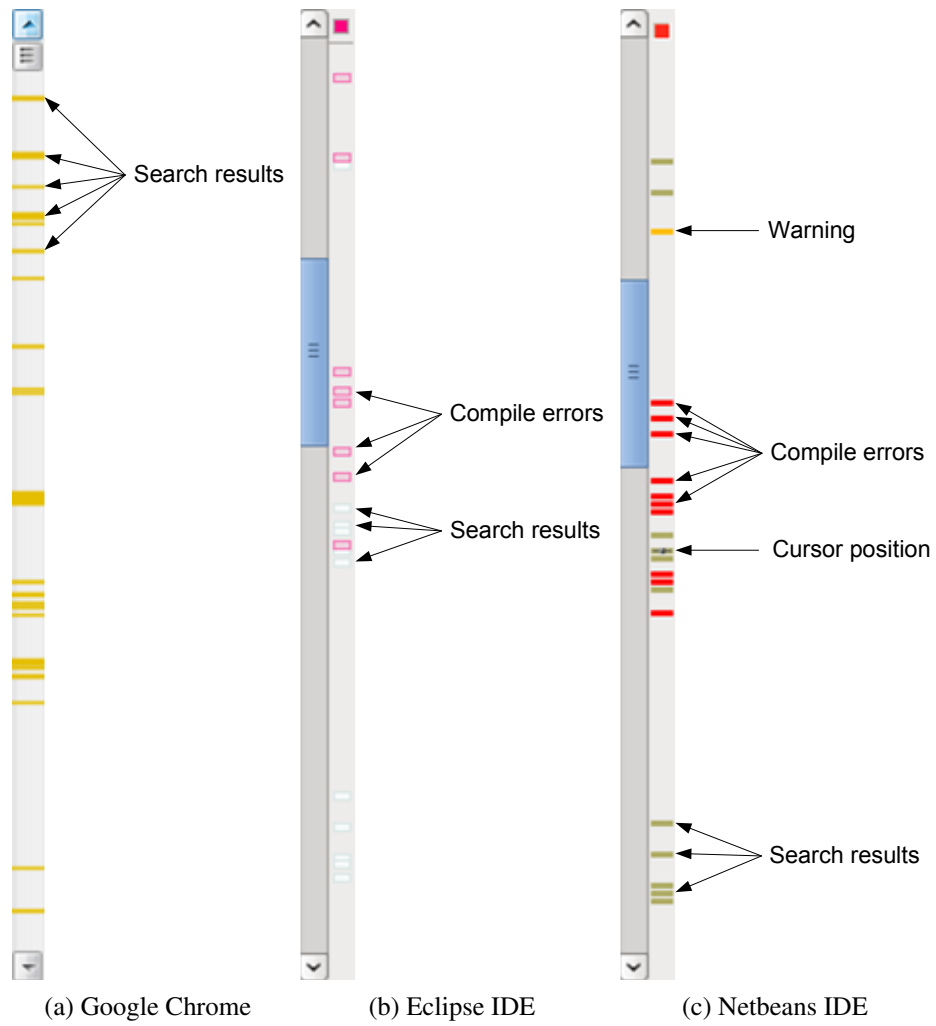


Figure 3.9: Scrollbar marks

pitch of a second tone, aiding position awareness. The tones at the beginning of the document have a higher pitch than those at the end. Both tones help the user to recognise when they encounter the situation of moving their mouse too far outside the scroll trough. To summarise, the first tone provides relative information (“you’re going up”) and the second provides absolute information (“you’re half way through the document”).

The auditory-scrollbar was significantly faster for a *navigate* task where participants were asked to go to a specific position in the document. Completion times for *search* tasks were not significantly different.

3.3.3 Scrollbar Behaviour Modifications

Scrollbars have a limited resolution: the thumb must maintain a minimum of several pixels in height, so it is visible and to allow acquisition. In long documents this means a single pixel movement in the thumb may alter the position by several pages. Traditional scrollbars overcome this issue by placing arrow buttons for small distance movement at each end, which is a “good solution in some cases” [7, pg. 366]. Two behavioural modifications to the scrollbar aim to address this issue, without the need for the scrollbar arrows.

Ahlberg and Shneiderman [7] suggested a split-thumb *Alphaslider* to allow more accurate position selection. The thumb is split along the axis of movement. Half of the thumb allows coarse movement, the other fine-grained movement, allowing users to first select an approximate position and then to easily refine it.

The *FineSlider* [146] uses a rubber-band metaphor for precise data selection on a scrollbar. The FineSlider has the same appearance as a traditional scrollbar, with the thumb operating in the normal manner when directly dragged. However, when the user clicks in the scrollbar trough, a rubber-band appears between the thumb and the click location. Instead of paging through the document, the rubber band pulls the thumb towards the selected location. A short rubber band (made by a click close to the thumb) initiates a slow movement, a long rubber band a faster movement. The rubber band mode of operation iterates over all content—this is especially useful in large documents where it is difficult to precisely position the thumb.

3.4 Content-Aware Navigation Aids

Content-aware navigation aids take advantage of their knowledge of the document's content to assist user navigation. Search and goto tools use their knowledge of the document to move the view to a specified position (see Section 3.1.5). Other tools that utilise document content for navigation are described in this section.

3.4.1 Content-Aware Scrolling (CAS)

Content-aware document movement requires a tool to have detailed knowledge of the underlying structure or the pertinent document positions. *Content Aware Scrolling* (CAS) [111] automatically varies the direction, speed and document zoom during a scroll action, based on the document content. For example, in a two column document, CAS automatically moves to the top of the right-hand column when reading of the left-hand column is complete.

The CAS widget contains two scrollbar thumbs. The first is unselectable, simply representing the document's position in the same manner as a standard scrollbar. The second is the CAS thumb. This is manipulated by the user to move through the document. The reason for requiring two thumbs is best demonstrated by an example. A user moves the CAS thumb vertically from the beginning of a single page, two column document, to the end. When the CAS thumb reaches half way, the standard thumb will be at the bottom of the trough (as the document will be at the bottom of the first column). As the user continues to drag the CAS thumb, the standard thumb will return to the top of the trough, as CAS has directed the browser to the top of the right-hand column. The two thumbs give the user an indication of both relative and absolute positions within the document.

CAS's content paths are automatically implied or user defined. CAS can also reduce user disorientation when panning between two search terms.

3.4.2 Intra-Document Links

Intra-document links provide indexes or bookmarks into salient areas of the document. These links are defined at creation time (such as tables of content) or by the document user (through bookmarking mechanisms) and support a non-linear

navigation path.

Links defined at creation time typically highlight structurally significant places within the document (chapter and section headings) or content that is parted from its description (tables and figures). Tables of content and indexes both provide links to pages within the document. Paper-based documents require the user to identify the correct entry by reading (or at least scanning) the list and then performing a visual search to locate the correct page. Electronic versions of the same documents may provide embedded links that take the reader to the correct page in the document with a single click, eliminating the visual search stage. The most common examples of internal links are the *# directive* used in web-pages (more commonly known as *hyperlinks*) and *Go-To Actions* used in PDF documents [4].

Footnotes and References

Footnotes and *references* link to extra content that is not part of the main text. Footnotes are commonly indicated by superscript numbers appearing in the document, requiring the reader to scan to the bottom of the page to locate the matching auxiliary text. Bibliographic references are often enclosed in round or square brackets and refer to a particular publication, listed in a bibliography or references section. Object references include those to tables or figures, or to related sections that are referenced from within the body of the document. Electronic documents may provide internal links to these features in a similar manner to tables of content and indexes; with a footnote, the page is simply scrolled to bring the footnote into a more comfortable reading position.

Bookmarks

Bookmarks are author, application or user defined, and mark pertinent document regions. Author defined bookmarks often link to chapter or section headings. Figure 3.10a illustrates Adobe Reader's mechanism for displaying such bookmarks. The pane of the left hand side lists text-based clickable bookmarks that link to various positions within the document. Application defined bookmarks are generated from the content the application is displaying. Figure 3.10b illustrates Visual Studio's drop-down list of the function definitions found in the open source code file. Clicking on a definition moves the document to that function. Figure 3.10b also

shows user-defined bookmarks, in a column to the left of the document. Users can bookmark any line and navigate the bookmarks by visual search or by using the functions provided in the toolbar. As an alternative, Ginsburg et al. [80] suggested using thumbnail trays to store bookmarks—a thumbnail of the bookmarked page is shown, instead of a text-based title. User-defined bookmarks are discussed further in Section 3.8.

Extremity Shortcuts

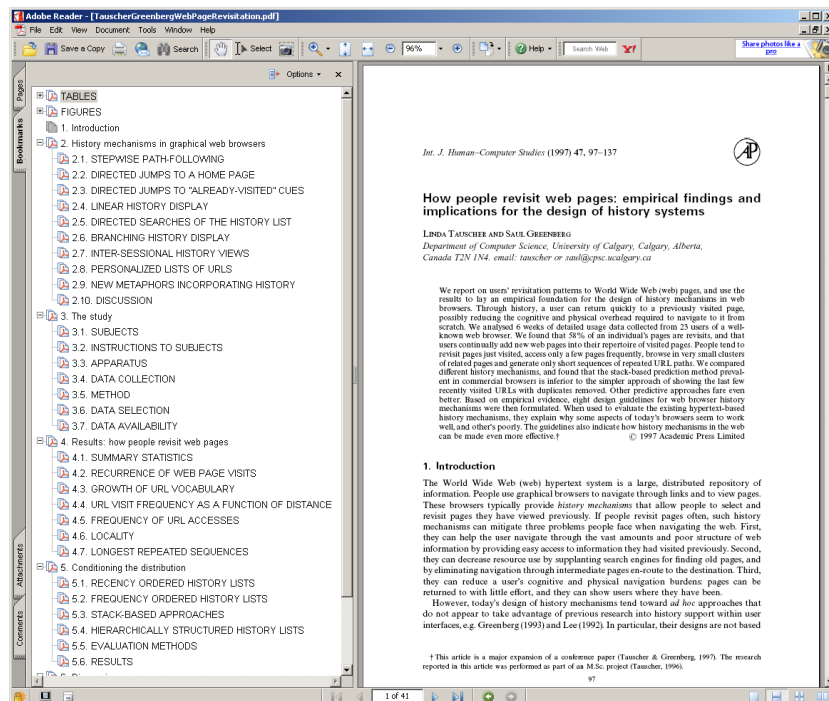
Navigation interfaces may simply use their knowledge of the document’s length to provide extremity shortcuts. It is common for navigation systems to include shortcuts to move to the first or last page in the document (Table 3.1, line 31). The Xpdf [76] application also provides shortcuts to move ten pages at a time (Table 3.1, line 32).

3.4.3 Semantic Browsing

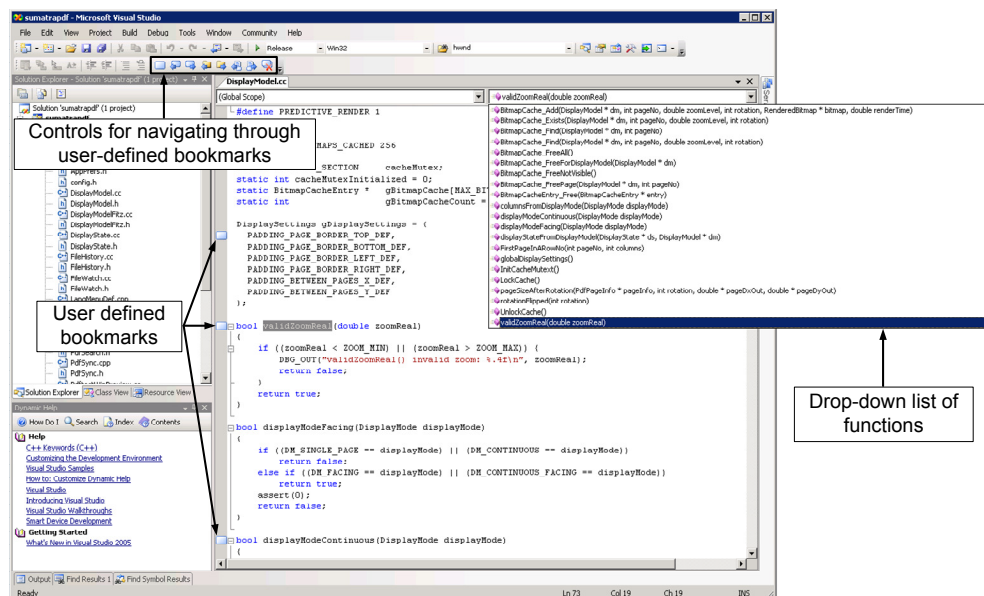
When browsing a document, a user may require access to a particular type of non-text content (for example a graph), but have little or no knowledge of its position within the document. *Semantic browsing* overcomes this issue by displaying or cycling through objects of a particular type. Figure 3.11 illustrates the semantic browse controls in Microsoft Word and Open Office Writer. To browse objects in Microsoft Word, the user presses the small circle below the scrollbar (Figure 3.11a) and selects the required object type (Figure 3.11b). The previous and next buttons (arrows in Figure 3.11a) are used to cycle through occurrences. Open Office Writer provides a listing of all object occurrences (Figure 3.11c), by category. Users expand the headings and select the object of interest. Examples of objects supported in these applications include sections, lines, bookmarks, footnotes, tables, figures, headings and comments. Other applications may support a range of further objects, depending on the document format and document content.

3.4.4 Content Visualisations

Content visualisations emphasise important document features to aid user navigation. These visualisations add to the standard document view or they may visu-



(a) Adobe Reader's bookmarks displayed in the left-hand pane



(b) Visual Studio's bookmarks. The application automatically creates the function bookmarks in the drop-down menu (top right). User defined bookmarks are placed in a column beside the source code (center left).

Figure 3.10: Example bookmarking mechanisms

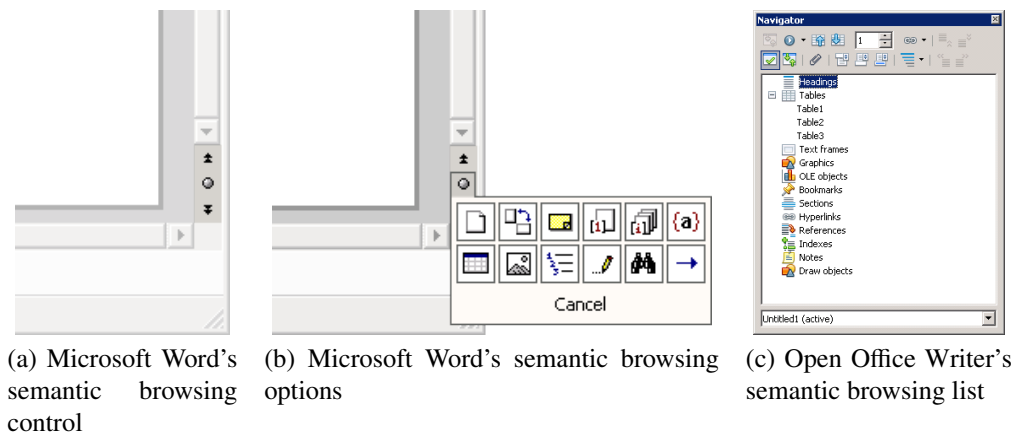


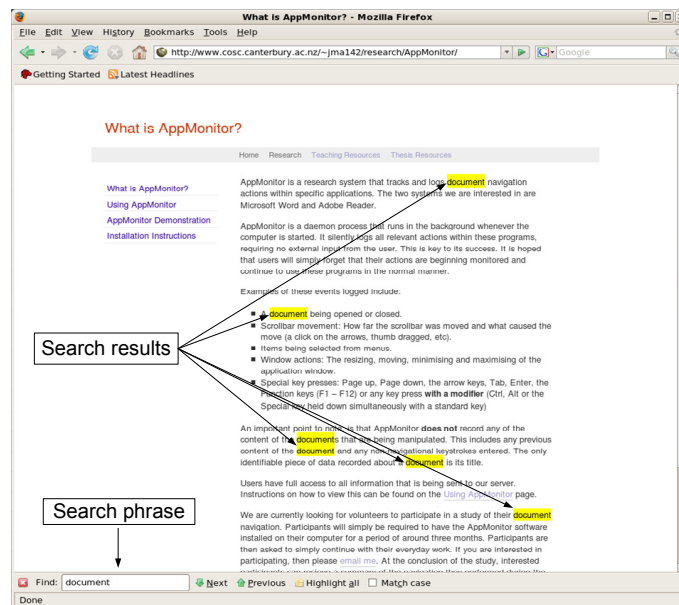
Figure 3.11: Semantic browsing controls

alise the content in a different manner. Four types of visualisations are described in this section: search aids, diff tools (to compare files), reviewing and multi-author tools, and aids for re-orientation after a scrolling action. Scrollbar-based content visualisations were described in Section 3.3.2.

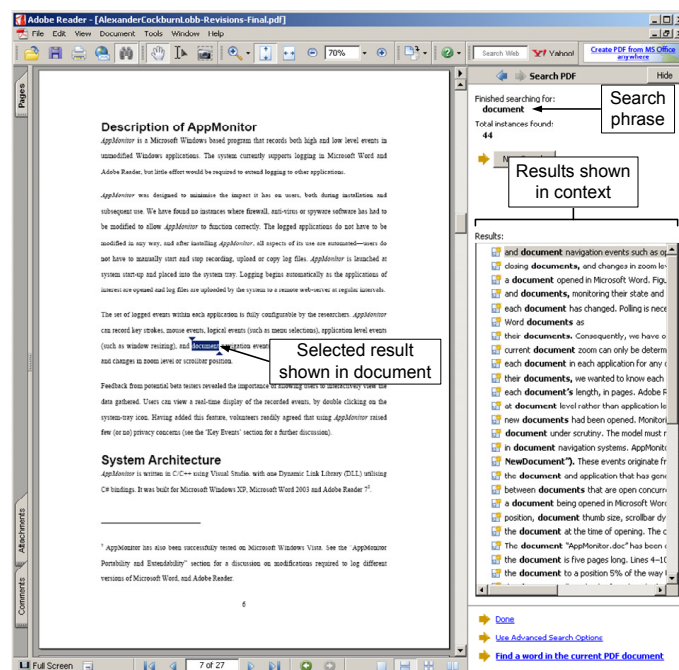
Search Aids

Many phrase search tools iterate through the results by jumping to the matching document position. This may disorientate the user, requiring them to read surrounding material to regain context. *Context-based search* addresses this issue by presenting all results simultaneously, in context, negating the need to jump between positions. This is an electronic form of *KeyWord In Context* (KWIC) commonly used in concordances. For example, the Mozilla Firefox browser [173] highlights the search results in the document (Figure 3.12a) and Adobe Reader generates a list of occurrences as well as the surrounding context (Figure 3.12b). Baudisch et al.'s Fishnet web browser [22] demonstrated the use of colour to highlight multiple search terms.

In large documents, there is the potential for a search phrase to occur numerous times throughout the document. The *Context Lens* [61] visualisation aims to globally visualise search terms within a document. It is a small rectangular widget that is placed in the top left hand corner of the interface. Rows represent sections



(a) Mozilla Firefox highlighted search results. The search phrase is typed in the box in the bottom left of the screen. Matches are highlighted in the web page as typing takes place.



(b) Adobe Reader's context-based search results. The search phrase is typed into the box at the top of the right-hand pane. The results, with context, are shown in the lower part of the same pane.

Figure 3.12: Context based search mechanisms

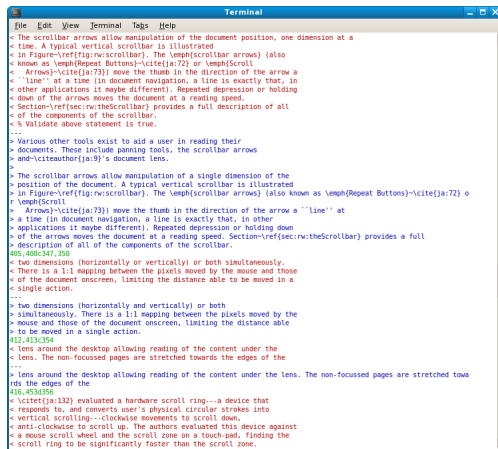
in the document; coloured squares in the columns represent the presence of that column's keyword in that section. Moving the mouse over a highlighted square pops up an in-context view of the keyword. It is not reported whether clicking on a section takes the user to that position within the document, however this is a potentially useful extension.

Semantically enhanced thumbnails provide a content overview as well as in-context results. Woodruff et al. [232] suggested *Enhanced Thumbnails* as an aid when scanning web pages in search engine results. Their thumbnails provide text and/or visual summaries of a web page, allowing the user to specify the highlighting applied to the search terms. Suh et al. [209] built on this idea with the *Popout Prism*. Their system uses perceptual principles to draw the user's attention to the search phrases. Words in a web page and thumbnail pane are enlarged and coloured causing them to "popout" from the normal text.

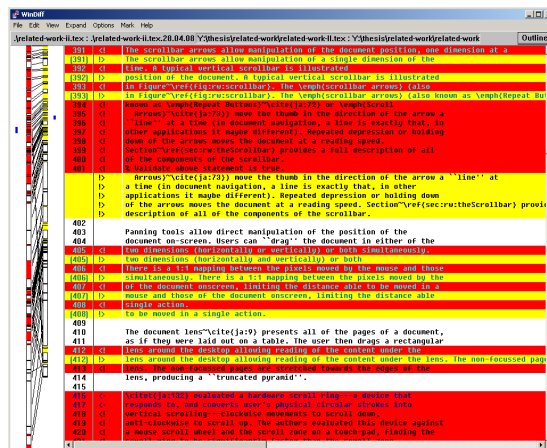
Diff Tools

Identifying the differences between two files is potentially a long and tedious process. *Diff* tools recognise and visualise these differences. A large number of implementations of these tools exist, most based on Hunt and McIlroy's [104] original algorithm. Diff tools are designed for working with plain text files—the visualisations are unlikely to be helpful for files with embedded binary data. *Diff* and *colordiff* (Figure 3.13a) are command line tools that indicate additions (a), deletions (d) and changes (c). Lines that are the same are not displayed. GUI based versions of these tools include *WinDiff* and *KDiff3* (Figures 3.13b and 3.13c respectively).

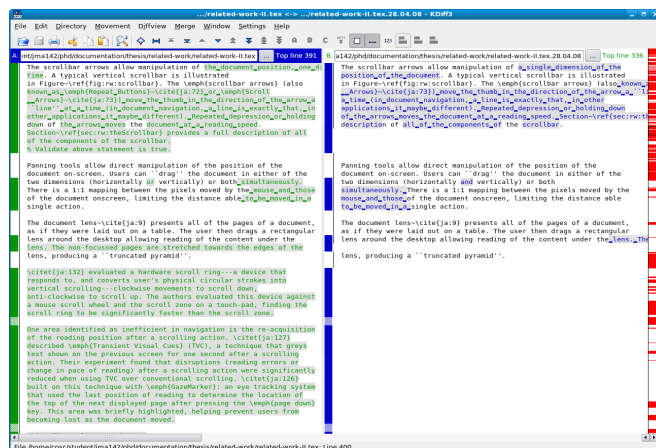
WinDiff [159] provides a visualisation on the left hand side of the interface summarising all of the changes between the two documents. One document is visualised in each column, with lines linking identical parts of the documents. Clicking in a document's column in the visualisation brings that part of the document into view. Differences from both documents are shown in the main pane, using different colours. In the primary visualisation, the lines with a red background are from the left hand document, lines with a yellow background are from the right-hand document. Lines that appear with a white background exist in both files.



(a) Color Diff



(b) WinDiff



(c) KDiff3

Figure 3.13: Diff tools

KDiff3 [67] shows the two files compared side-by-side, with an overview of the differences shown in a bar, to the left of the standard scrollbar. Clicking in this bar takes the user directly to the selected portion of the comparison.

Reviewing and Multi-Authoring

Reviewing and multi-author tools visualise different authors' modifications to the document. These tools may work serially or in parallel. Serial visualisations show the authors' individual modifications, with the document then passed to the next person. Parallel visualisations show the current modification positions of other authors who are simultaneously working on the same content.

Serial document reviewing tools allow tracking of document versions, insertions and deletions to be traced, viewed, accepted or rejected, and suggested edits and comments to be added. Figure 3.14 illustrates the appearance of these tools in Microsoft Word. Insertions into the document are indicated by red-text and a thin bar in the right-hand margin (①, ⑤); deletions by a bar in the margin and a text bubble displaying the removed text (④). Formatting changes are indicated by a bubble (②) and suggested edits and comments are shown in shaded balloons (③). These visualisations easily allow the reader to identify modified areas of the document or areas where attention is required. Changes made by different authors are identified by different colours and by their initials in the comment balloons. Microsoft Word provides functions for moving between edits; for example, the *next* and *previous* functions on the reviewing toolbar (⑥). Other applications provide similar tools.

Collaborative (or parallel) authoring and reviewing tools visualise the changes that multiple authors are making, simultaneously, to the same document. These tools use the idea of a *shared workspace*, onto which each author has a view that they can modify. Most collaborative authoring widgets are based around awareness of other authors in the document. *Radar views* [94] allow users to visualise where in the document others are editing. Squares show the area displayed on another author's screen and *telepointers* [20, 94] represent their mouse pointer. Baecker et al.'s SASE and SASSE [20] collaborative authoring environments include colour coded text sections (to allow locking) and multiple scrollbars. A vertical scrollbar is added for each collaborator on the document, the rightmost

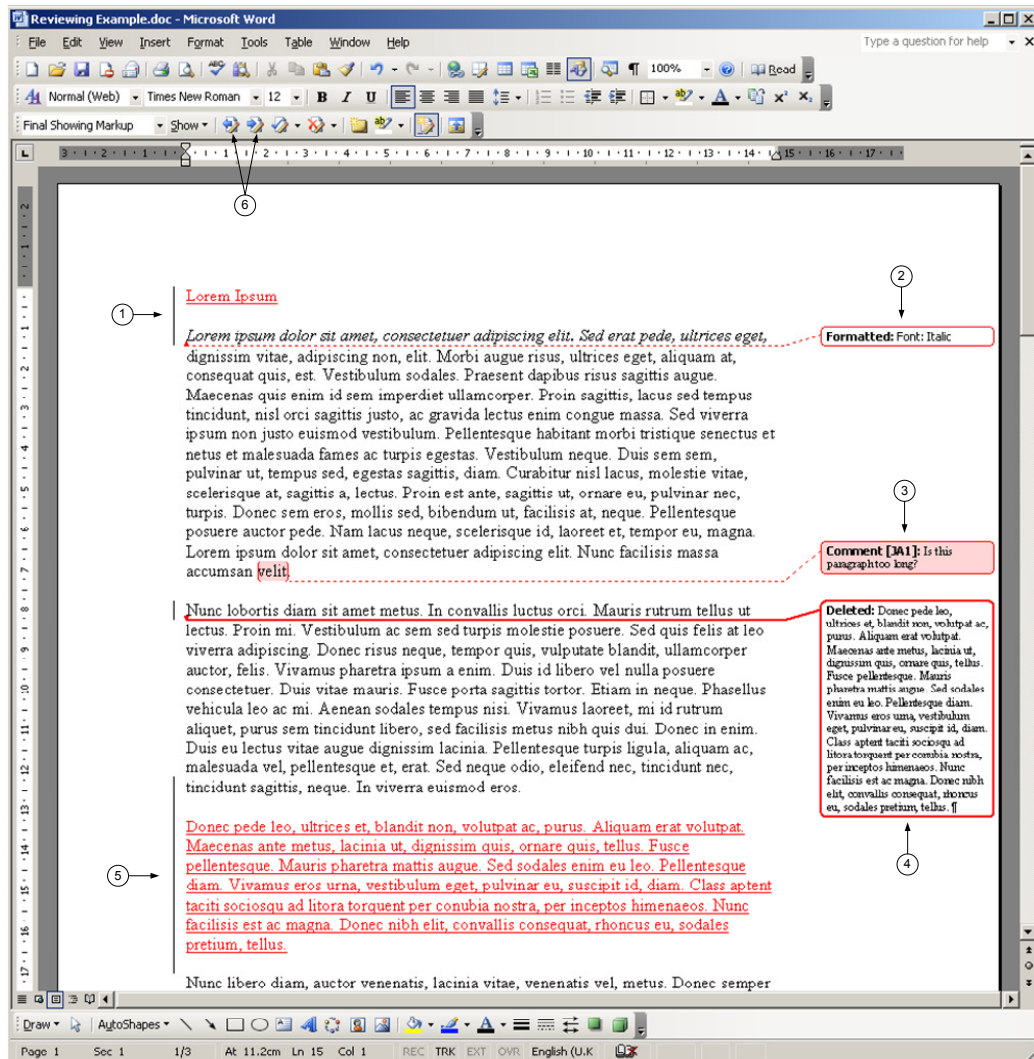


Figure 3.14: Microsoft Word's reviewing tools. ① and ⑤: insertions, ②: content formatting, ③: comments, ④: deleted text, ⑥: previous and next edit tools.

belonging to the user and the remaining (read-only) scrollbars indicating the positions of other authors within the document. Collaborative authoring is aided by locking the scrollbars together, giving a WYSIWIS⁴ environment. Tools such as *Anchored Conversations* [46] allow conversations between authors to be “anchored” to a particular point in the document, allowing easy, directed discussion, especially when authors are a long distance apart.

Re-orientation After a Scrolling Action

When a document is paged or scrolled, the new view may also display a few lines of the previous view. This helps users to relate the newly displayed content to that which it has just replaced. However, this means that the old content must be scanned before reaching the new, which may be inefficient if performed regularly. Kaptelinin et al. [118] described *Transient Visual Cues* (TVC), a technique that grays text shown on the previous screen for one second after a scrolling action. Their experiment found that disruptions (reading errors or changes in pace of reading) after a scrolling action were significantly reduced when using TVC over conventional scrolling. Kumar and Winograd [128] built on this technique with *GazeMarker*, an eye tracking system that used the last position of reading to determine the location of the top of the next displayed page, triggered by pressing the *page down* key. This area was briefly highlighted, helping to prevent users from becoming lost as the document moved.

Klein and Bederson [127] investigated the benefits of animations when moving between different views of a document. They found that the addition of animations significantly improves reading time and decreases error rates. Animations of 300 ms were optimal for reading tasks. The authors suggest the benefit of having animations outweighs the small amount of extra time spent in the animation stage.

3.5 Document Visualisations that Provide Multiple Views

Multi-view techniques provide two or more views onto the same document. Advances in screen technology mean larger displays are readily available, providing sufficient space to display multiple simultaneous views onto a document.

⁴What You See Is What I See

3.5.1 *Split-Views*

Split-view interfaces allow multiple, simultaneous views onto a single document. These are particularly useful for comparison tasks or when re-arranging content. Split-views are differentiated from techniques such as TES (where thumbnails are provided down one column of the document, see Section 3.3.1), by regarding a split-view interface as one where two or more *readable* views onto the same document are provided. Microsoft Word’s implementation of a split view interface is shown in Figure 3.15. The top viewport scrolls and zooms independently from the bottom. This tool is also common in spreadsheet applications such as Microsoft Excel [160] and Open Office Calc [183].

3.5.2 *Overview+Detail*

Overview+detail interfaces simultaneously provide two distinct views of the information space. One provides a zoomed in, detailed visualisation of a small portion of this space, the other a less detailed view of the entire document. This type of visualisation is useful to users of large documents, providing context to their detailed view. One of the simplest but most common types of overview+detail interface is the scrollbar, providing a one-dimensional overview of the document [55]. Scrollbars and their enhancements were described in Section 3.1.1 and Section 3.3. Two example overview+detail interfaces are shown in Figure 3.16. Figure 3.16a shows the overview+detail interface for the *Google Maps* [130] browser and Figure 3.16b the pan and zoom window in Adobe Reader 9 [5].

The *Thumbnail Slider* [101] is a tool that places a single thumbnail of the current document page in one corner of the screen. The thumbnail image is dragged right or left to cycle the page images forward or backward in an RSVP like manner. The mouse button is released when the target page is located, updating the main display.

Overview+detail interfaces may also use Z-separation to distinguish multiple views [55]. Lenses use Z-separation to magnify or augment a region of the view. Document previewers, such as Xdvi [222] and Yap (which is part of the MiKTeX typesetting package [197]) contain magnification lenses that are activated when holding down a mouse button.

Overviews may also be implemented at the same size as the detail view. Ma-

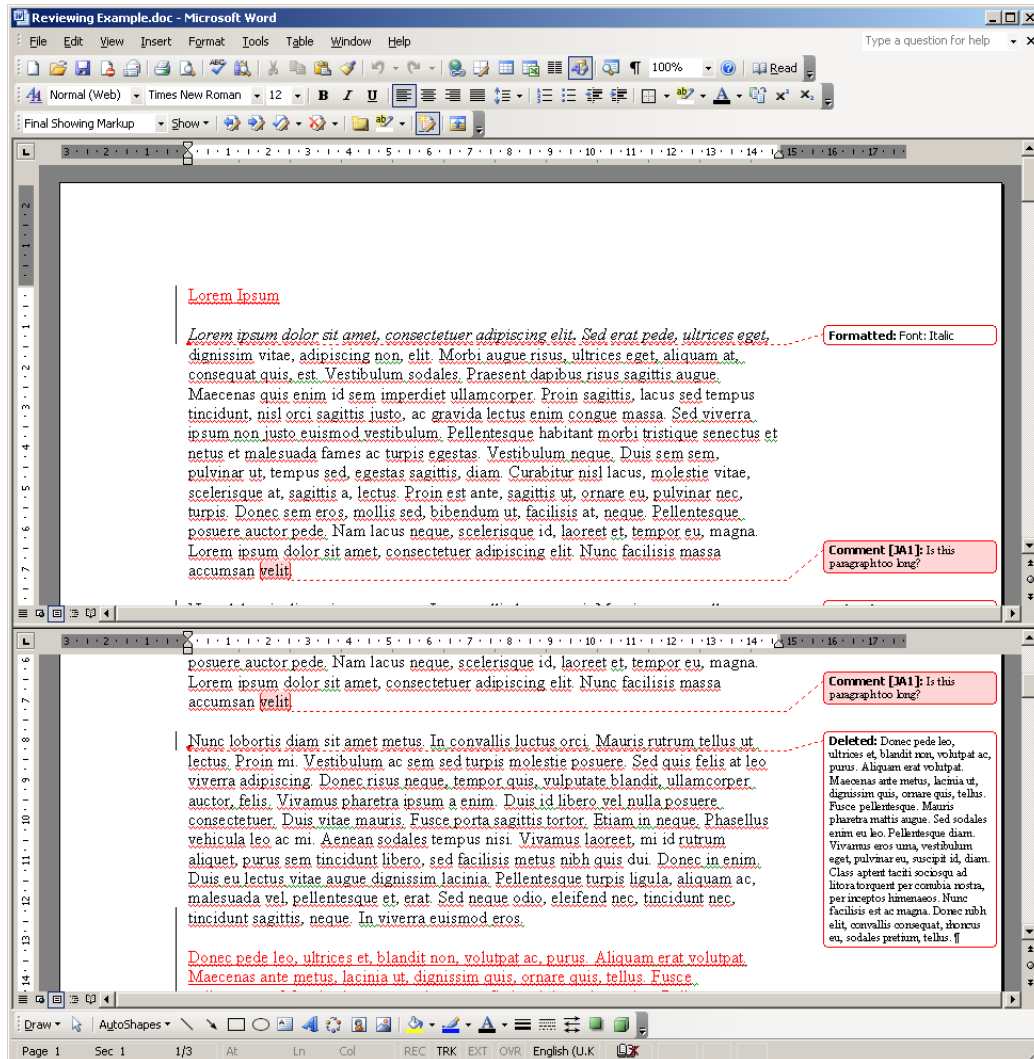
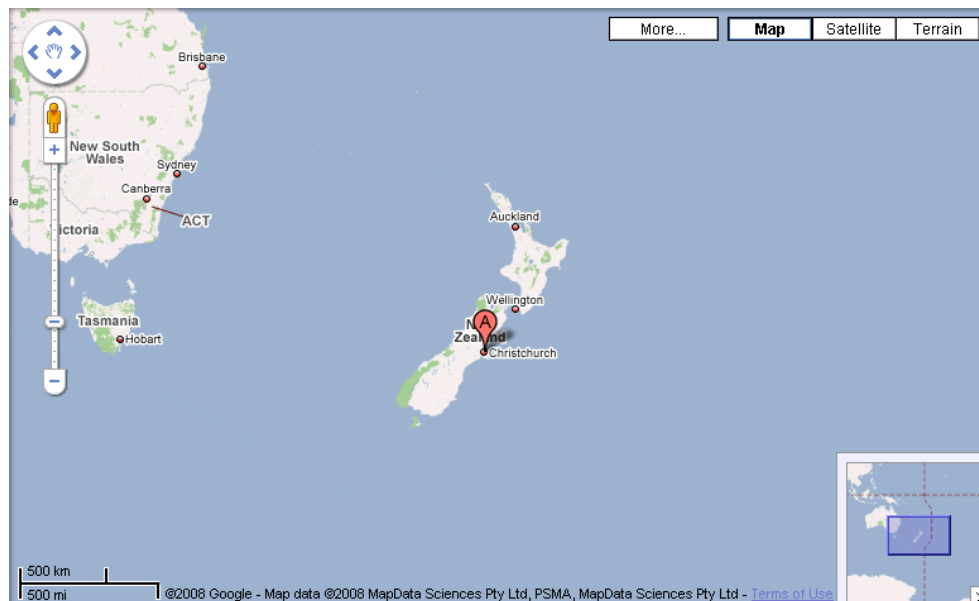
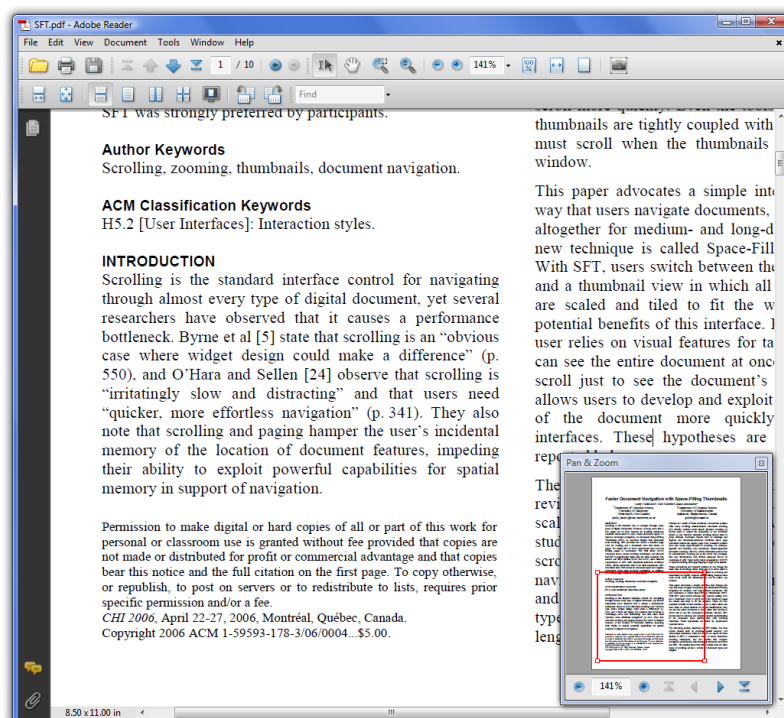


Figure 3.15: Microsoft Word's split-view interface. The top and bottom views support independent navigation.



(a) Google Maps overview+detail map browser. The overview is shown in the bottom right corner. Dragging the rectangle inside the overview panel adjusts the detailed view.



(b) Adobe Reader's pan and zoom interface. The overview panel, currently in the bottom right corner, is resizeable and moveable anywhere within or outside the main window. Moving or resizing the box in the overview panel modifies the detail view.

Figure 3.16: Example overview+detail interfaces

soodian et al. [145] noted that users of word processor applications lack contextual information. Their DeepDocument system used a multi-layer LCD screen to simultaneously display multiple views of a document to ease this problem. The rear layer displayed thumbnails of the pages in the document (the overview), and the front layer displayed the document at normal reading zoom (the detail). This system has the advantage that it does not spatially separate the overview and the detail—something that can lead to users neglecting the overview [57].

3.5.3 *Focus+Context*

Focus+context interfaces overcome the view separation issue observed in overview+detail interfaces, by displaying the focus “seamlessly within its surrounding context” [55]. A common method for achieving this visualisation is through fisheye techniques. Fisheye views display large information spaces by providing a “balance of local detail and global context” [79, pg. 16]. Furnas [79] first proposed and evaluated a fisheye interface for source code—details at the edges of the view were traded for higher-level contextual information. The fisheye view was superior to a flat view of a file for navigation tasks that involved moving from the current location to a target. Code-folding interfaces that allow programmers to ‘fold up’ methods using ‘+’ and ‘−’ controls are an example of a manual fisheye that achieve a similar result.

Since Furnas’s fisheye view, many other researchers have implemented fish-eyes for various tasks. The *Document Lens* [193] presents all of the pages of a document, as if they were laid out on a table. The user then drags a rectangular lens around the desktop to read the document. The non-focused pages are stretched towards the edges of the lens, producing a truncated pyramid effect. Hornbæk and Frøkjær [103] created a fisheye interface for reading electronic documents. More important portions of a document are always visible. Less important regions are distorted, becoming unreadable, but are expanded when clicked. The authors conducted an experiment that involved essay writing and answering questions about the document the participants had read. They found that subjects completed their essays faster using the fisheye interface, but gained a less complete understanding of the text.

Baudisch et al.’s [22] Fishnet browser displayed all of a web-page’s content on

one screen, using a fisheye view to show a portion of the page in detail, with the remainder spatially compressed. Jakobsen and Hornbæk applied a fisheye view to source code that displayed “those parts of the source code that have the highest degree of interest given the current focus” [112, pg. 377].

Finally, Björk [25] introduced *Hierarchical Flip Zooming*, a focus+context interface that places contextual object tiles around a larger object of focus. Tiles are placed in sequential order around the primary object. Potentially, each tile could represent a page in a document.

3.6 Navigation by Indirect Manipulation

The majority of the interface techniques presented in this chapter use direct manipulation to control document navigation. However, several techniques exist that use do not use this interaction paradigm, including rate-controlled techniques, navigation using rotational actions, two-handed navigation, perspective view navigation and navigation using eye-tracking. Each of these techniques are described in this section.

3.6.1 Rate-Controlled Techniques

Rate control systems allow the user to control the *velocity* of movement instead of the (more common) document position. *Rate-Based Scrolling* is the most widely applied rate-control technique and was described in Section 3.1.4. Two further techniques used rate control: Speed-dependent Automatic Zooming, that couples document zoom to scroll speed, and page flipping.

Speed-Dependent Automatic Zooming (SDAZ)

Rate-based scrolling allows very rapid movement through long documents. However, users may suffer motion blur as pages quickly speed into and out of view. In an attempt to reduce this issue, Igarashi and Hinckley [109] introduced Speed-Dependent Automatic Zooming (SDAZ). SDAZ is a rate-controlled system that ties the zoom level of the document to the velocity of movement—the faster the document is scrolled, the further it is zoomed out.

Cockburn and Savage [50] evaluated SDAZ for both document and map based tasks. SDAZ was significantly faster than traditional scroll, pan and zoom techniques for map browsing and also significantly faster than scrollbars for document-based tasks. In their work on tuning the parameters of SDAZ, Cockburn et al. [52] evaluated SDAZ, scrollbars, rate-based scrolling and Displacement Dependent Automatic Zooming (DDAZ)⁵. For visual search tasks, the authors found that calibrated SDAZ outperformed all other scrolling techniques.

Page-Flipping

Paging allows the user to move through a document, one page at a time. The single page on-screen is replaced by the next or previous page (see Section 3.1.3). *Page-flipping* is the rapid application of the *paging* technique—document pages are presented in quick succession, exploiting the human’s ability to quickly recognise distinctive images.

Rapid Serial Visual Presentation (RSVP) techniques expose the user to a series of pages from a document in quick succession. This is analogous to a person flipping through the pages in a book. Previous applications of RSVP techniques include: to sets of images (Spence et al. [207]), to streams of words (Boguraev et al. [27], Öquist and Lundin [184]) and to document thumbnails (Lawton and Feigin [131]). Sun and Guimbretière’s Flipper [210] was the first system to *explicitly* apply RSVP to full sized document pages (note that a scrollbar can imitate RSVP by clicking and holding in the scroll trough).

Flipper is a rate-controlled system that at slow scroll speeds utilises SDAZ [109] to control position and document zoom. At faster speeds, the document is zoomed to a single page on-screen and RSVP is employed to rapidly display the document content. Flipper also implements a backtracking algorithm to account for the difference in document position between when a target is recognised and when the scroll speed is set to zero to halt the page flipping. Experimental analysis found Flipper was significantly faster than TES and SDAZ. Unfortunately, users sometimes found the backtracking confusing and disorienting.

To isolate the performance advantages of an RSVP system, Cockburn et al.

⁵DDAZ: “after each discrete thumb movement a pan/zoom animation quickly moves between the original and new document locations” [52, pg. 76].

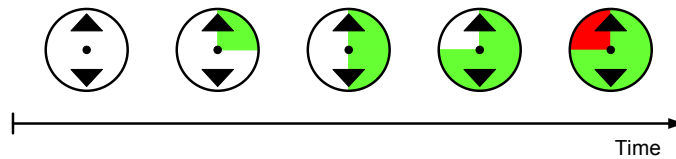


Figure 3.17: RSVP rate control icon giving visual feedback on the next page flip

[53] evaluated two versions of RSVP against five other document navigation systems. The first version, RSVP, displayed a single page of the document on the screen at a time. The second system, Multi-page Rapid Serial Visual Presentation (MRSVP), was equivalent to Spence et al.’s [207] “mixed-mode” presentation, displaying four images at a time. Users moved between the blocks of four pages using a rate-control technique. To view one of the four pages in a block, the user clicked on it.

Both of their systems allowed the user to control the rate of “flipping” in the same manner as a traditional rate-control system—the further the cursor is moved from the initial position, the faster the document scrolls. The authors limited page flipping to a maximum rate of 10 pages/sec [53]. Both RSVP and MRSVP provided the user with a gradually filling rate-based scrolling icon, as illustrated in Figure 3.17. Starting empty as a new page is shown, the cursor icon fills, proportional to the speed of movement through the document. Once full, the next (or previous, depending on scrolling direction) page in the document is displayed and the cursor emptied.

Cockburn et al.’s evaluation of document navigation systems found MRSVP to be the second most efficient system for both visual search and spatial relocation tasks. RSVP performed approximately the same as traditional scrollbars for the visual search tasks, but worse in the spatial relocate tasks. However, RSVP systems were uniformly disliked, inducing comments such as “it felt like I was about to get a headache ... or a seizure”.

3.6.2 Navigation Using Rotational Actions

The idea of using rotational movements to modify the document view has existed since the first GUI interfaces, when Sutherland [212] used dials to change the x-

and y- co-ordinates of the display of a drawing. This method of navigation was largely ignored until 2003 when Wherry described the *Hardware Scroll Ring*.

Scroll Rings

A *hardware scroll ring* converts physical circular strokes into vertical scrolling—clockwise movements to scroll down, anti-clockwise to scroll up. Wherry [228] used Fitts' law tasks to compare the performance of a touchpad scroll zone, the mousewheel and the hardware scroll ring. The scroll ring performed better than all of the other tools.

The hardware scroll ring, like any device, has an associated purchase cost. To overcome this cost, Moscovich and Hughes [172] translated the hardware scroll ring into a software based scroll ring. The *Virtual Scroll Ring* (VSR) allows the user to make circular movements with their normal input device (the mouse) to scroll the document. These circles are unconstrained, meaning their sizes (and shape) are variable. To compensate for this, VSR uses the distance travelled along the circumference of the circle drawn, instead of the angle swept, as the input for the distance to scroll. Small slow movements produce slow scrolling, large fast movements produce fast scrolling. The author's evaluation found that VSR performed at least as well as a standard mouse wheel for medium and long distances. However, the mousewheel was preferred for short distances as it was more accurate.

The Radial Scroll Tool

Acquiring a scrollbar widget on large or small scale touch-screen is potentially a difficult task. On large screens the user may have to physically move their whole body to reach the widget. On small screens the limited display space means only a small proportion is assigned to navigation widgets. To resolve this issue, Smith and Schraefel [203] described the *radial scroll tool*. It allows the user to gesture on a touch-sensitive surface and move in a clockwise or anti-clockwise direction to navigate within the document. The initial gesture triggers a transparent overlay that shows a number of radial lines from a central point. Moving clockwise over the lines moves the document forward, anti-clockwise backwards. Increasing the distance from the centre point increases the distance the document moves.

The radial scroll tool requires that the circular gestures are centered around the initial gesture position. However, if the user drifts from the central point, movement through the document is no longer smooth. To resolve this issue, Smith and Schraefel [203] suggested *Curve Dial*, a technique that records the curvature of the gesture instead of the crossing of radial lines. This allows smooth scrolling even when the gestures are not perfectly circular. Importantly, it removes the need for the user to concentrate their attention on the gestures, making curve-dial an eyes-free navigation technique.

3.6.3 Two-handed Navigation

Two hands are used for nearly all of the tasks performed in everyday life. Buxton [35] argued that two-handed interaction should be applied to computer-based navigation in the same way. Using an information display point-of-view, a GUI is *space multiplexed*—different objects for different functions are displayed simultaneously. However, from the control perspective, majority of input is made using a single input device (the mouse) and is *time multiplexed*, leading to serial input. Further, direct manipulation tasks are classified as either *foreground* tasks (performed with the dominant hand) or *background* tasks (performed with the non-dominant hand). Document navigation tasks should be performed as a background task, using the non-dominant hand, leaving the dominant hand to continue with the primary task. Buxton suggests three actions on touch-sensitive surfaces for navigation: smooth scrolling (touch-slide-release), page turning (touch-rapid flick-release) and jumping to location (touch-release).

Two-handed interaction is significantly faster for tasks that involve navigation/selection and scrolling/pointing. Buxton and Myers [34] experimentally showed that both novices and experts were able to perform ‘navigate and select’ tasks faster with two-handed than with single-handed input. In their evaluation of input devices for scrolling and pointing tasks, Zhai et al. [236] used a bi-manual technique as one of their experimental conditions. Participants controlled scrolling actions using a TrackPoint isometric joystick mounted in the keyboard and used a mouse for pointing tasks. For visual search tasks, they found this combination was significantly faster than using the scrollbar or mousewheel independently. Finally, Myers et al. [174] used a PDA and mouse as input for a scrolling task. The

PDA displayed line up, line down, page up and page down buttons, a slider (where sliding the finger slid the document by the same amount), rate control input and an absolute control input. The mouse was used for pointing. The authors found two-handed input with the mouse and PDA can match or beat other single-handed and two-handed techniques.

3.6.4 *Perspective View Navigation*

Traditional presentation methods provide a camera view that is perpendicular to the document's surface. This arrangement is good for reading tasks, but prevents users from looking ahead (or behind) when navigating.

Perspective View (PV) techniques challenge this primary visualisation, suggesting that documents should be shown with a tilted camera, providing one solution to the focus+context problem [93]. Guiard et al. [92] describe three types of camera rotations that can form perspective views:

Panoramic rotation: The camera rotates from a fixed point above the document.

Lunar rotation: The camera is moved along a semi-circle, maintaining a constant fixation point in the document (akin to the moon rotating around the earth).

Trans-rotation: The camera moves along a line parallel to the document's surface, but remains fixated to a constant point.

Guiard et al. [92] compared perspective view navigation with pan-and-zoom (P&Z) techniques. For target acquisition tasks, the PV technique exhibited a concave-up curvature, while P&Z adhered to Fitts' law [93]. For tasks with Indexes of Difficulty (ID) greater than 15 bits, P&Z techniques performed better, for IDs less than 9 bits PV was faster.

3.6.5 *Navigation Using Eye-Tracking*

All systems described thus far have required users to *explicitly* indicate when a navigation action is to take place. Kumar and Winograd [128] proposed three automatic scrolling techniques that detect from a user's eye-gaze when the document view should be modified. The *eye-in-the-middle* technique dynamically

adjusts the scrolling speed of the document to maintain the user's gaze in the middle third of the screen. *Smooth scrolling with gaze-repositioning* automatically scrolls (at slightly faster than reading pace) the document when the user's eye gaze falls below (an invisible) threshold line on-screen. If the eye-gaze falls further towards the bottom of the screen, the scrolling speed increases. Scrolling stops when the eye-gaze returns to a threshold area near the top of the screen. Finally, *discrete scrolling with gaze-repositioning* calls a page-down function using *GazeMarker* when the user's eye falls below a threshold on the screen. The authors also implemented off-screen buttons for traditional discrete and continuous scrolling functions that were activated by staring at targets positioned in space.

3.7 Zoom Tools

Zoom tools allow the user to modify the on-screen magnification of the document. The larger the zoom value, the “closer” the document appears to the screen. Manual zoom control forms a core aspect of document navigation and was described in Section 3.1.7. Advanced zooming tools such as semantic zooming (modifying the content as the zoom changes), parallel scale manipulation (where zooming is performed in parallel with another task) and automatic zooming (the zoom level is automatically modified due to another action) are described in this section.

3.7.1 Semantic Zooming

Most zoom functionality provides an enlarged or reduced version of the current document view. Semantic zooming modifies the content displayed according to the zoom level. Pad [186] and Pad++ [23] demonstrated this technique by modifying the type of information displayed about an object as the magnification changed. For example, when zoomed out, a calendar may only show years. When partially zoomed in, it may show months; days and times only appear after repeated zoom actions. Woodruff et al. [232] used a form of semantic zooming in their enhanced thumbnails, highlighting important features in results returned from a search engine. Suh et al.'s [209] popout-prism semantically zoomed search results on a web page (see Section 3.4.4 for more details on enhanced thumbnails and the popout-prism).

3.7.2 *Parallel Scale Manipulation*

Two-handed input decreased task time by allowing the user to perform two tasks simultaneously; parallel scale manipulation allows the user to perform two tasks simultaneously with single-handed input. In this context, the zoom level of a document is manipulated at the same time as performing another task. *Zliding* and *Multi-scale Navigation* both support this type of navigation.

Zliding (or Zoom Sliding) allows the user to control both the zoom and a linear parameter (possibly document position) simultaneously. This is achieved using a pressure sensitive stylus—pressure controls the zoom, while the x-y cursor position controls parameter selection. The Zlider widget looks similar to a scrollbar. Moving the stylus over the widget with low pressure allows course granularity of movement, zooming in by applying pressure allows more precise movement or selection [188].

Multi-scale navigation tasks involve large 1D or 2D environments where panning and zooming are used, simultaneously, to acquire a target. Multi-scale target acquisition tasks consist of three parts: zooming out, zooming in while panning and pointing to the target with the cursor [90]. Input is made through a mouse or stylus; a stylus should be used if greater precision is required [89]. Multi-scale navigation is most efficient when the panning and zooming tasks are performed in parallel [28, 29]. Much of the research in this area is based on abstract pointing tasks; however, Appert and Fekete proposed *OrthoZoom Scroller*—a multi-scale document navigation technique.

OrthoZoom Scroller “requires only the mouse to perform panning and zooming in a 1D space” [11, pg. 21]. Vertical scrolling is performed in the normal manner. Zooming in is achieved by dragging orthogonal to the surface of the scrollbar. The authors found that the OrthoZoom Scroller was “about twice as fast” [11, pg. 26] as SDAZ for a pointing task. A Multi-Scale Table of Contents (MSTOC) was suggested for browsing large documents, semantically adding and removing entries as the zoom changed.

With the large number of multi-scale navigation techniques proposed, Guiard et al. [91] suggested using Shakespeare’s complete works as a benchmark for evaluating multi-scale navigation in the context of document navigation. This is described further in Section 3.10.2.

3.7.3 Automatic Zooming

Automatic zooming interfaces adjust the document magnification as a consequence of another user action. There are two examples of this type of interface: Speed-Dependent Automatic Zooming (SDAZ), as described in Section 3.6.1, and Space-Filling Thumbnails.

Cockburn et al. [53] argued that for efficient visual search and spatial re-acquisition, scrollbars should be eliminated. To satisfy this requirement, they introduced Space-Filling Thumbnails (SFT). Their design lays out, in a matrix, a thumbnail image of every page of the document, on a single screen. The thumbnails are scaled appropriately to prevent the need for scrollbars. The user clicks on one of the thumbnails to zoom in to a reading-level magnification; they return to the matrix view in the same manner. In an initial study, SFT was found to be significantly faster than seven other document navigation interfaces. A more detailed experiment found that SFT performed “faster than TES across all document types and lengths” [53, pg. 1] and was efficient and subjectively preferred for both visual search and revisitation tasks.

3.8 Revisitation Tools

Returning to a previously viewed document position is potentially a frustrating task, especially when the user can visualise the required content, but must spend a large amount of time searching for its position within the document.

Revisitation tools aid the user in quickly returning to positions of interest. While many navigation tools aid this task to some extent (for example, recalling from the scrollbar thumb that the section was about “half way through the document”), revisitation tools give *explicit* support for such actions. The majority of the tools described here provide automatic mechanisms for maintaining the list of positions visited; only *user-defined bookmarks* must be explicitly created by the user.

3.8.1 User-Defined Bookmarks

Bookmarking a document position (possibly with a label) allows for quick re-acquisition of that location. In 1992 Shneiderman [199] suggested several im-

provements to the scrollbar, including the ability to mark particular positions on the scrollbar and the ability to return to those positions by simply clicking.

Building on Shneiderman's idea of marking areas in the scroll trough, Laakso et al. [129] introduced the *Bookmark Scrollbar*. The Bookmark Scrollbar provides an area next to the standard scrollbar into which the user can add bookmarks (with labels). Dragging the scroll thumb into the bookmark area forces it to snap to the closest bookmark and move the document view to that position. As an alternative, Ginsburg et al. [80] allowed users to drag whole pages into a thumbnail tray as the bookmarking mechanism in their *DeckView* system.

3.8.2 *Electronic Dog-Ears*

Dog-ears are commonly observed in paper-back books; accidental dog-ears are a sign of use, while explicit dog-ears mark important pages. Electronic dog-ears provide the same information. To encode the frequency of visits to a web-page, Cockburn et al.'s [51] *WebView* placed a dog-ear in the top left corner of page thumbnails. The dog-ear encodes information regarding the number of visits to that page—the darker the 'ear', the greater the number of previous visits to that page. To bookmark particular pages, users can explicitly add a dog-ear to the bottom left corner of the thumbnail.

In traditional RSVP displays, each image stays on-screen for the same amount of time. Hoeben and Stappers [101] added dog-ears to their Thumbnail Sliders (see Section 3.5.2) to alter the length of time each RSVP thumbnail stayed visible. Heavy dog-ears indicated important pages that stayed on screen for longer than lightly or un-dog-eared pages. The weight of the bookmark could be changed manually or encoded from the document content.

3.8.3 *Click-Through Navigation*

Click-through navigation is designed to emulate the physical activity of retaining positions within a book by inserting a finger a specific point [31]. Users begin by selecting a piece of content they wish to return to at a later point in time. A snapshot of this position is taken. This snapshot appears when the mouse has paused for a short duration and then follows the pointer around the screen. Navigation continues, but when the user wishes to return to the selected document position,

they simply double-click the snapshot. The document moves to that location and the snapshot is replaced by one of the previous document location.

Document figures are often separated from the text that references them. Readers may wish to quickly refer to the figure and return to their position in the document. Buchanan and Owen [33] implemented *Figure Navigation*, a type of click-through navigation that addresses this issue. When a reference to a figure in a document is encountered, a thumbnail of the related figure appears (it is not reported *when* the thumbnail appears—on mouseover of the figure reference or if it is always visible). Clicking on the thumbnail takes the user to that position in the document. A floating palette allows users to quickly return the position in the text from which the figure was referenced.

3.8.4 *Previous View and Next View*

A user's view onto a document consists of a vertical position, a horizontal position, a zoom level and an orientation. By maintaining a history of the changes to a document's view, applications can assist users to quickly return to these positions. The *previous view* function, as the name suggests, allows the user to return to their previous document view. After this action, the *next view* function will navigate forward through the history, to return to the original view. An application's view history may support some or all of the parameters that define a view.

3.9 *Navigation Techniques on Mobile Devices*

The widespread use of mobile devices, such as phones and PDAs, has encouraged researchers to improve the navigation mechanisms for handheld interfaces. Interaction with these devices poses unique challenges, with limited screen real-estate and alternate input channels to those available on a traditional desktop computer. However, techniques designed for these devices could provide valuable lessons or ideas for desktop-based document navigation. This section looks at proposed navigation techniques for document navigation on mobile devices.

3.9.1 *Small-Screen Devices*

Improvements to document navigation on small-screen devices have focused on modifying the interface (finding replacements for scrollbars) and modifying the input method (using physical motions and actions to control on-screen content).

Improving the Interface

Interface improvements include RSVP and flip-zooming as reading aids, *wiping* for re-orientation after a scroll action and position/pressure as a control for scrolling. Lawton and Feigin [131] described *Streaming Thumbnails*, a reading technique that presents a low-resolution thumbnail image of a complete document page on screen. When the user clicks on a portion of the text they wish to read, the system displays a box, inside which the text is streamed via RSVP. Björk and Redström [26] ported flip-zooming [25] (Section 3.5.3) to a small screen device. Short, three word summaries and hyperlinks are placed into the on-screen tiles.

Wiping [150] is a method for quick re-orientation after scrolling on a small-screen device. When a paging action is initiated, all of the old text is grayed. The new (black) text is scrolled up to replace the old, guiding the user's eye to the reading position. *Touch-n-Go* allows users to apply pressure to a touch-screen to begin scrolling. Direction is controlled by the position of the touch from the center of the screen, scrolling speed by the pressure with which it is applied [59].

Improving the Input Method

The primary input method for most devices with small-screens is either a numeric-keypad or a touch screen; however, the small physical dimensions make device manipulation another option for navigation input. Tilting the device is a viable alternative input method. Harrison et al. [96] added a tilt sensor to a Palm Pilot Handheld device using tilts as input to the rate of movement through a list; Es-lambolchilar and Murray-Smith [70] implemented tilt-based SDAZ on a handheld device.

However, tilting “limits the usefulness of the visual display” [178, pg. 317] during input and so non-visual vibrotactile feedback can aid navigation. Rate control is represented by modifying the frequency of feedback, position control by altering one dimension of the vibrotactile feedback, and contextual feedback is

based on the content scrolled [178]. Eslambolchilar et al. [71] later added auditory feedback to their tilt-based SDAZ. In a similar vain, Bartlett's *Rock 'n' Scroll* [21], maps physical camera rotations to photo viewing functions.

The small physical size of these devices has encouraged researchers to investigate using an *object in the world* paradigm instead of the traditional *world of objects* for navigation [72]. Displays showing a 'world of objects' look down onto the information space. The view is changed by manipulating the position of the 'world'. In contrast, 'object in the world' displays take on the role of an object. Physical device manipulation reveals new information—in this case, it is the object that is moved, not the world. This technique was tested on flat surfaces [72], using a device's camera and the background environment [171], using an additional marker system held behind the device to control position and zoom [95] and by adding the mechanics of a mouse on the reverse side of the device for table-top and hand-based use [200].

3.9.2 *E-book Readers*

E-book readers are portable hardware devices specifically designed for displaying electronic documents. They are smaller, lighter and consume less power than laptop computers, but have a large enough screen to comfortably display a page of a book for reading. Some e-book readers provide *bi-stable* displays that do not require power in order to display a constant image on-screen (and allow reading under bright light) [43]. Commercial examples of e-book readers include the Amazon Kindle [9] and the Sony Reader [206]. These devices allow users to view PDF files, plain text files, image files and variety of other book formats. The Kindle supports bookmarking, highlighting and search functions. Revisitation is supported by returning to the last reading position when a document is reopened.

In an effort to more realistically emulate traditional books, Chen et al. [42, 43] presented *dual-display* e-book readers. These provide users with two displays instead of the more standard single screen. The dual displays are used attached (representing an open book) or detached to support a variety of interactions common in paper documents. These interactions include folding, flipping, fanning and flexible layout on a desk.

3.10 Evaluating Document Navigation Tools and Techniques

Unlike more mature areas of Human-Computer Interaction, document navigation researchers have no standard methodology for evaluating and comparing new tools and techniques. There are several reasons for this lack of standardisation: the vast number of tasks requiring navigation (see Chapter 2), user familiarity with the document, preciseness of the target, the previous task, the location of the user's previous action, the current position of the user's hands on input devices and the ease with which new tools are learned. With such a large number of factors influencing the use of navigation tools, document navigation techniques are usually evaluated in small, targeted studies, to isolate performance differences. This section investigates the methodologies and results of studies that have evaluated document navigation tools.

3.10.1 What to Measure?

Both quantitative (empirical) and qualitative (subjective) measures should be considered when evaluating any new system. Table 3.3 provides a survey of the literature that report document navigation tool evaluations and summarises the measures employed by researchers. The survey is presented as evaluation/author pairs, with the evaluation column detailing the primary tool under study. *Study type* is reported as *formal* (F) or *informal* (I). Formal studies are those that report experimental setup and conditions in enough detail to allow reproducibility. The empirical measures are divided into measures of task time, error rate (also a measure of accuracy) and other empirical results. The subjective measures are divided into the use of NASA-TLX surveys, preference rankings and other subjective questions. Note that casual observations are often also made by the experimenters, leading to important insights. These are not recorded in this table.

Task completion time is the most regularly employed empirical performance measure and preference the most regularly evaluated subjective measure. All formal studies report task completion time and the majority also report preference statistics. Informal studies tend to only report a single quantitative or qualitative measure.

Several studies only employed either an empirical measure *or* a subjective measure. While one type of evaluation may seem acceptable ("the quickest one

Evaluation	Author (s)	Study type	Empirical measures			Subjective measures		
			Task time	Error rate	Other	Preference	NASA-TLX	Other
Auditory enhanced SB	Brewster et al. [30]	F	✓	✓		✓	✓	✓
Animated scrolling	Klein et al. [127]	F	✓	✓		✓		✓
Animation	Donskoy et al. [65]	F	✓			✓		
Bookmark scrollbar	Laakso et al. [129]	I						✓
Code thumbnails	DeLine et al. [60]	F	✓			✓		✓
Confetti scrollbar	Byrd [36]	F	✓	✓		✓		✓
Electronic dog ears	Hoeben et al. [101]	I			✓			
Eye-gaze scrolling	Kumar et al. [128]	I				✓		
FineSlider	Masui et al. [146]	I	✓					
Fisheye views	Baudisch et al. [22]	F	✓	✓		✓		
	Hornbæk et al. [103]	F	✓			✓		✓
	Jakobsen et al. [112]	F	✓	✓		✓		✓
Flipper	Sun et al. [210]	F	✓			✓	✓	
GazeMarker	Kumar et al. [128]	I				✓		
Radial scroll tool	Smith et al. [203]	F	✓			✓		
Hardware scroll ring	Wherry [228]	F	✓	✓		✓		
Mousewheel	Hinckley et al. [100]	F	✓	✓	✓	✓		
	Ren et al. [190]	F	✓					✓
	Zhai et al. [236]	F	✓					✓
Mural bars	McCrickard et al. [148]	F	✓		✓			
OrthoZoom scroller	Appert et al. [11]	F	✓	✓	✓	✓		
SDAZ	Cockburn et al. [50]	F	✓			✓	✓	
	Cockburn et al. [52]	F	✓			✓	✓	
	Igarashi et al. [109]	I	✓			✓		
	Savage et al. [196]	F	✓			✓	✓	
SlideBar	Chipman et al. [45]	F	✓			✓		✓
SFT	Cockburn et al. [53]	F	✓			✓	✓	
Thumbnail sliders	Hoeben et al. [101]	I						✓
TVC	Kaptelinin et al. [118]	F	✓	✓				
Virtual scroll ring	Moscovich et al. [172]	F	✓			✓		✓

Table 3.3: Methods for evaluating document navigation tool performance. *Study Type* is reported as formal (F) or informal (I). Formal studies are those that report experimental setup and conditions in enough detail to allow reproducibility.

must be the best” or “people would choose this one, so it must be the best”), a good example of the need for at least one measure from both categories is provided by Cockburn et al.’s evaluations [53]. After a pilot study, the selection of interfaces for a “head-to-head” study was swayed by subjective opinions, after disturbing comments from subjects regarding one of the fastest interfaces. The authors took both measures into account, meaning the selection of interfaces for further investigation was not purely based on task completion time.

To aid the evaluation of multiscale navigation techniques, Guiard et al. [93] offered *Degree of Goal Directedness* (DGD) as a measure of the cumulative pay-off for reaching a target within a document. Most evaluations require users to navigate to a specific point in a document to complete a task. DGD describes the usefulness of the information the user gains while travelling to the specified document position. For tasks that have a low DGD value (the information gained along the way *is* important) researchers can use measures of the ease with which the information is obtained or the amount of information obtained as an additional evaluation criteria.

3.10.2 Evaluation Frameworks

Evaluation frameworks describe a methodology for comparing document navigation tools. They detail *how* the factors in the previous section should be measured. No such framework is regularly employed by all document navigation researchers; instead, many studies are conducted using ‘ad hoc’ methods to test specific interactions. In some cases, the standardised methodologies of Fitts’ law evaluations were employed to measure the performance of document navigation systems.

The ISO9241 [66] standard outlines evaluation procedures for measuring performance, comfort and effort using Fitts’ law [74] (see Section 3.2.1). While originally used to measure a tapping task between two static points, document navigation researchers have shown that Fitts’ law also holds for scrolling movements [100]. Since Hinckley et al.’s [100] work, several researchers have employed simple Fitts’ law tasks for measuring the efficiency of document navigation systems (see Table 3.4, line 38).

Guiard et al. [91] note that most Fitts’ law tasks use abstract targets on a blank background—an environment quite different to that of an electronic document. In-

stead, the authors offer Shakespeare’s complete works as a benchmark for evaluating multi-scale document navigation techniques. Their framework supports pan-and-zoom, SDAZ, OrthoZoom and perspective view navigation. Shakespeare’s 37 plays, which equate to approximately 150,000 lines, are arranged in a single column HTML document. Selecting a single verse from this corpus equates to a Fitts’ law task with an approximate difficulty of 17.2 bits.

Later, Guiard et al. [93] proposed the *Serial Target Acquisition* (STA) paradigm as an extension to typical Fitts’ law tasks. In an STA task, the participant must visualise (rather than select) multiple targets as they move serially through a document. This correlates to document navigation tasks such as checking the formatting of tables or figures. Fitts’ law modelling can be applied to each distinct target, as each has a known distance and width.

3.10.3 *Evaluations of Document Navigation Tools and Techniques*

Numerous researchers have evaluated document navigation tools using a variety of evaluation strategies, types of documents, document lengths and numbers of participants. These evaluations are summarised in Table 3.4. Studies are labelled as formal or informal, as reported in Table 3.3. The majority of studies are formally conducted, providing full details on the experimental method. The systems evaluated in each study are shown via tick marks, with those that were overall quantitatively or qualitatively ‘the best’ noted (these designations are somewhat crude—it is impossible to summarise complete experimental outcomes using elementary marks).

The most evaluated document navigation tool is the scrollbar. The widespread deployment of this tool means researchers wish to prove their systems against the current de facto standard. Only one of evaluations show the scrollbar to be either empirically or subjectively superior, yet it remains a cornerstone of virtually all GUI toolkits. Many experiments have also evaluated the mousewheel and SDAZ (Table 3.4, lines 12 and 26).

Visual search is the most commonly employed task type (Table 3.4, line 44) and is over-represented compared with the other document navigation tasks identified in Chapter 2. Potentially, this is because visual search tasks are easy to evaluate or because visual search is a commonly performed task (although the

literature does not validate this proposition). Even with their wide deployment, visual search tasks are still quite variable; researchers must decide how the task should be cued (by an image, text or audio), whether the direction and/or distance to the target is given, the starting position within the document and the accuracy required to achieve the task.

The document types and lengths used for evaluations are quite variant (Table 3.4, lines 46–65), as is the participant base for testing (line 66). This is likely due to the lack of knowledge as to the documents used by people in their everyday work environments and specific experimental requirements. Only Cockburn et al. [53] and McCrickard and Catrambone [148] report studies that involve more than 30 participants.

3.10.4 Modeling Tool Performance

Models of navigation tool performance predict the time it will take to complete a task. Several researchers have constructed models of document navigation tools, either as their main goal or as a side-product of tool evaluations. In some cases (such as for scrolling) contention exists between authors as to the correct model.

Generalised and Stationary Scrolling

As a preliminary to modelling tool performance, it is important to accurately describe the display location of objects in abstract spaces. Smith and Taivalsaari [204] proposed such a model. Equation 3.2 illustrates their *generalised model of scrolling*, where x is an object's display location, f is a scrolling function, x_A is an object's absolute position in an N-dimensional space and p is the scrolling offset. Both f and p are vectors so can affect multiple aspects of display location.

$$x = f(x_A - p) \quad (3.2)$$

The authors applied their model to several variations of scrolling, including linear, radial, fisheye and pan & zoom movement. To illustrate the application of their model, the linear model of scrolling is now described. Consider a plain text file. The display location, x , is simply the onscreen co-ordinates of the character, (x, y) . The position's offset, p , is the vertical value taken from the scrollbar. The

		Evaluation († = multiple evaluations reported)																											
		Appert et al. [11]	Baudisch et al. [22]	Brewster et al. [30]	Byrd [36]	Chipman et al. [45]	Cockburn et al. [50]	Cockburn et al. [52]	Cockburn et al. [53] †	DeLine et al. [60]	Donskoy et al. [65]	Hinckley et al. [100]	Hoeben et al. [101] †	Hombek et al. [103]	Igarashi et al. [109]	Jakobsen et al. [112]	Kaptein et al. [118]	Klein et al. [127]	Kumar et al. [128] †	Masui et al. [146]	McCrickard et al. [148]	Moscowich et al. [172]	Ren et al. [190] †	Savage et al. [196]	Smith et al. [203]	Sun et al. [210]	Wherry [228]	Zhai et al. [236]	
1	Study type	F	F	F	F	F	F	F	F	F	F	F	I	F	I	F	F	F	F	I	I	F	F	F	F	F	F	F	
Evaluation systems (☑ = quantitatively the ‘best’ system, ☒ = qualitatively the ‘best’ system, ☑☒ = quantitatively and qualitatively ‘best’ system)																													
2	Animation																												
3	Auditory enhanced scrollbar		☑☒																										
4	Code thumbnails																												
5	Confetti scrollbar				☒																								
6	Context-based search																												
7	Electronic dog ears																												
8	Eye-gaze scrolling																			☑									
9	Flipper																												
10	FineSlider																										☑☒		
11	Fisheye views		☑											☒		☑													
12	GazeMarker																												
13	Mousewheel																												
14	Mousewheel with acceleration																												
15	Mural bar and pile bar																												
16	OrthoZoom scroller	☑																											
17	Overview+detail		☒											☑☒															
18	Pan+zoom																												
19	Radial scroll tool																												
20	Paging																												
21	Rate-control																												
22	RSVP																												
23	MRSVP																												
24	Scroll ring																												
25	Scrollbar																												
26	Scrollpoint																												
27	SDAZ								☑	☑☒															☑☒	☑☒			
28	SFT								☑☒	☑☒															☑☒				
29	SlideBar								☑☒																				
30	TES																												
31	Thumbnail sliders																												
32	Transient visual cues																												
33	Two-handed input																												
34	Touch-pad scroll zone																												
35	Virtual scroll ring																												
36	Zooming																												
37	Other																											☑	
Task type																													
38	Count																												
39	Fitts’ law																												
40	Free form																												
41	Goto																												
42	Reading																												
43	Re-acquisition																												
44	Relevance																												
45	Visual search																												
46	Not reported																												
Document																													
47	Abstract targets																												
48	Book																												
49	Conference paper																												
50	Formatted text																												
51	Journal paper																												
52	List																												
53	Manual																												
54	Plain text																												
55	Source code																												
56	Symbols																												
57	Thesis																												
58	Web pages																												
59	Other																												
60	Not reported																												
Document length																													
61	Items																												
62	Lines																												
63	Pages																												
64	Pixels																												
65	Words																												
66	Not reported																												
67	Number of participants	12	13	12	6	24	12	27	13, 32, 14	11	12	27	5, NR	20	7	16	10	20	10	8	76	8	12	3	8	12	12	12	

Table 3.4: Summary of document navigation tool evaluations

absolute position of the character within the document, $x_A = (x_A, y_A)$, always remains constant, regardless of scrolling actions. This model for vertical scrolling is shown in Equation 3.3. By increasing the scrollbar position, p , the displayed y value for the object decreases.

$$\begin{aligned}(x, y) &= f(x_A - p) \\ &= (x_A, y_A) - (0, p)\end{aligned}\tag{3.3}$$

Interested readers should see Smith and Taivalsaari’s [204] work for details of their models for other navigation techniques.

Smith and Taivalsaari also describe a model called *Stationary Scrolling*, where all objects remain stationary on screen and “scrolling” is based on the saliency of an object. Each object has a non-spatial quantity associated with it (such as opacity or size), as scrolling occurs, it is this value which is modified, instead of spatial location.

Target Acquisition

Two primary models exist that describe how fast a target is reached in a document: Fitts’ law models and a linear model. Hinckley et al. [100] used a reciprocal framing task to evaluate ScrollPoint, and a mousewheel with and without acceleration, for scrolling through documents. For their task, they found that Fitts’ law does model each of their devices, with correlations between 0.9–0.97 for movement time vs. index of difficulty plots. Participants in this study always knew the direction (and approximate distance) to the target, as tasks were cued by line numbers that were also displayed beside the text. This represented scrolling in a familiar document. Savage and Cockburn [196] also found their results of acquiring off-screen targets adhered to Fitts’ law. In their tasks, a green arrow in the center of the screen constantly displayed the direction to the target, meaning its location was to some degree “known”.

In comparison, Andersen [10] proposed a linear movement time model for scrolling when the position of the target is not known ahead of time and the data is not sorted. Andersen argued that scrolling in these types of documents is limited by the physical time it takes to move a pointer or other control *and* by the maximum speed at which visual scanning can take place. This implies a maximum

speed plateau, which is not observed in acquisition tasks that obey Fitts' law [74] (which follow a typical bell shaped curve). This model was experimentally validated.

The lack of models for other task types is indicative of the number of factors that influence navigation.

3.11 Summary and Discussion

This chapter has presented a review of electronic document navigation tools and their associated evaluations. A common core of techniques are widely deployed in current applications. Improvements to these tools fall into seven categories: input devices, scrollbar augmentations, content-aware navigation aids, visualisations that provide multiple views, indirect manipulation techniques, zooming tools and revisitation tools. No standard methodology for evaluating document navigation tools is currently in widespread use—past evaluations have applied a range of tasks, documents, and qualitative and quantitative measures to examine and compare the efficiency of new techniques.

3.11.1 Adoption of Improvements into Document Navigation Systems

The adoption and implementation of improvements to tools in widely used navigation systems is generally low. Several factors influence this lack of support: the expense associated with purchasing new hardware devices, the cost or complexity of implementing new techniques, interference with existing tools, the small benefits available for large learning curves, a lack of proof of these benefits and the immaturity of new techniques.

The mouse and keyboard have dominated input to computers since personal computing became readily and inexpensively available. For new input devices to be universally adopted, they must be cheap, have significant benefits over current input and have multiple uses. Mass production will usually reduce costs (although only after many users have already shown interest), however the final two issues will require a device design with clear benefits coupled with marketing and package deals by influential companies. Augmentations to current devices—for example the mousewheel—may have a greater chance of success than dedicated document navigation devices.

The cost or complexity of implementation or incompatibility with current tools may influence the decision to support software-based navigation techniques. The tools presented in this section range from simple but worthwhile improvements to major conceptual shifts in presentation. Designers should consider two aspects of cost: the cost to the user (for example of new hardware) and the cost of implementation. Several of the techniques described require specialist hardware (such as eye-tracking devices) or more powerful hardware (such as graphics cards) that are not provided on many low-end computers. Users are unlikely to buy expensive hardware unless the benefits are clear. Implementation (and testing) costs are tied to the complexity of the new tool. This may be dependent on the architecture of the current system and the ease of integration of improvements. Large companies are unlikely to have issues implementing complex tools, while open source initiatives, where developers work for little or no pay, may have insufficient resources to implement these techniques.

The system architecture also dictates whether new tools are compatible with current tools. For example, if a scrollbar is rendered with edit wear and read wear, users may become confused by the additional rendering of search results in the same trough. Screen real-estate may also dictate that alternatives are impractical. Further, designers are often reluctant to change interface behaviour that already exists, for fear of confusing current users.

The immaturity of the tool and the lack of research into its benefits is potentially the single biggest reason for *not* supporting a new technique. Many of the tools described in this chapter never make it out of research labs. It is unlikely that designers of document navigation systems would seriously consider tools proposed by a single research group and never further studied by other researchers.

Users will only benefit from new document navigation tools when they are included in the applications they use everyday. However, to justify inclusion, designers must be sure their cost and effort will produce significant gains for the user. User adoption of the new tool is influenced by the effort they have to put in to learn it—if the new technique it is hard to learn or to use, it is unlikely users will switch. It is the job of researchers to provide compelling systems along with consistent, reproducible evaluations, to prove the worth of their tools to the communities that design document navigation systems.

3.11.2 Issues Associated with Document Navigation Tool Evaluations

The final section in this chapter explored evaluations of document navigation systems. The measures commonly employed, frameworks to aid consistent evaluation, a survey of evaluations of document navigation tools and the resulting models of performance were discussed. Consistent and repeatable evaluations of tools are essential to convince designers that new techniques outperform those currently in use and that the cost and effort of implementation will benefit their users. The primary issue that arose from this exploration is the lack of a standardised and realistic testing framework, resulting in a lack of fair comparisons and models of performance.

Fitts' law tasks in scrolling environments has arisen as one suitable avenue for testing target acquisition times. This technique is useful for testing the optimal performance of target acquisition, but is a somewhat artificial document navigation task. These evaluations are often performed in blank documents, possibly with concentric circles around the two targets. Real world document navigation is performed in non-blank documents with targets that are rarely as defined. Guiard et al.'s [91] framework using Shakespeare's complete works is one step towards a more realistic context.

Visual search tasks are also heavily represented in evaluations of document navigation systems. The current lack of knowledge as to the extent the tasks outlined in Chapter 2 are performed, requires researchers to estimate their importance for inclusion in evaluations. It remains to be seen whether Fitts' law and visual search tasks are summarative of 'real world' document navigation actions.

Part II

Characterising Electronic Document Navigation

Chapter 4

AppMonitor: A Tool for Recording Document Navigation Actions in Unmodified Windows Applications

USER interface (UI) designers would benefit from a strong empirical understanding of users' document navigation actions. This would provide them with insights into common user behaviours and areas of inefficiency. The previous two chapters describe several studies that observed paper-based document navigation and many studies that evaluated electronic document navigation tools under laboratory conditions. Field studies have reported some quantitative data on the use of electronic document navigation tools, but only tangentially as part of other research. To help fill this knowledge gap, this chapter describes *AppMonitor*, a tool that unobtrusively logs user actions in unmodified Windows applications. The following chapters describe studies that employed AppMonitor to record users' actions during a longitudinal study and a task-centric observation.

AppMonitor is a client-side logging tool that records user actions in Microsoft Word and Adobe Reader. AppMonitor uses the Windows SDK libraries to monitor both low level interactions such as "left mouse button pressed" and "Ctrl-f pressed" as well as high level 'logical' actions such as menu selections and scrollbar manipulations. No user input is required to manage logging, allowing participants to seamlessly conduct their everyday work while their actions are observed. AppMonitor logs events in Microsoft Word and Adobe Reader; however, its architecture allows extension to any Windows-based application.

This chapter describes the motivation, design and output from AppMonitor. This will help other researchers who wish to use or extend AppMonitor or deploy similar logging software. The chapter begins by outlining the motivation for creating an application logger, followed by a description of the system requirements.

It then discusses the AppMonitor architecture, starting with a high level overview that is followed by details of the lower level components. Next, the steps involved in porting AppMonitor to other versions of the Windows operating system and the procedures for extending it to monitor other applications are explained. The major issues encountered during implementation are then presented, giving pointers for other researchers creating similar systems. Following this, details on how researchers can configure the AppMonitor system are provided, along with a description of the log files AppMonitor produces. Finally, the methods employed when parsing and analysing the log files are discussed, again giving pointers to other researchers analysing similar types of data sets.

4.1 Motivation for Creating an Application Logger

Application loggers allow accurate recording of user actions in an unobtrusive manner. Their primary advantage over human-observation techniques is that they allow longitudinal studies of days, weeks, months or even years of real-world use (as discussed in Section 2.3.2). Studies that require human observers are impractical for these long periods of time and those that involve video- or screen-recordings are time-consuming to analyse. In contrast, the structured output from application loggers can be parsed and analysed mechanically. Unobtrusive application loggers also reduce the Hawthorne Effect (the phenomenon observed when participants under study modify their behaviour) [147]. A series of well-defined system requirements were created to guide the construction of this tool.

4.2 System Requirements

The primary goal of AppMonitor is to unobtrusively record document navigation actions in Microsoft Word and Adobe Reader. A set of system requirements were drawn up to ensure AppMonitor would be suitable for the longitudinal and task-centric studies that were planned. These are enumerated in this section. AppMonitor would:

- 1. Log users' navigation actions in Microsoft Word and Adobe Reader.*

This goal summaries the primary objective of AppMonitor—to record events that will allow empirical analysis of document navigation actions. Mi-

Microsoft Word and Adobe Reader were chosen due to their widespread use on Windows-based desktop computers. The Microsoft Office 2007 suite sold more than 120 million licenses in the 2008 fiscal year [229] and Adobe Reader's website indicates that "500 million copies have been distributed worldwide since 1993" [5]. Recruiting participants who were familiar with these applications and who regularly used them would be easier than if less common applications were selected. Two applications were chosen to ensure the behaviours recorded were not unique to a particular system.

2. *Allow users to continue with their everyday work using unmodified versions of their normal applications. Software substitutes were not to be applied.*

This is pertinent to the longitudinal study that is described in Chapter 5. Many studies have asked participants to use custom versions of familiar applications or open-source alternatives to allow logging to take place (see Section 2.3.2). The disadvantage of this approach is that users may modify their behaviour as they become familiar with the different or new interface. AppMonitor would be built to avoid this issue, by logging *unmodified* versions of these applications.

3. *Perform this logging with minimal system performance degradation.*

If participants suffer performance degradation of their computer while under study, they will be less likely to participate for long periods of time.

4. *Require no input from users to begin or end logging, or manage log files.*

This goal stipulates that users will not be required to perform any administrative functions during the study. This helps to prevent intrusion on their work and interference by the Hawthorne Effect.

5. *Allow users to view the events recorded.*

This helps to ease any user concerns regarding the actions AppMonitor is recording during the study.

6. *Allow logging to be manually interrupted, if required.*

These six high-level requirements guided the design and development of AppMonitor.

4.3 System Architecture

AppMonitor is written in C/C++ using Microsoft Visual Studio [154], with one Dynamic Link Library (DLL) utilising C# bindings. It was built to run on Microsoft Windows XP [155] or Vista [169] and to monitor user actions in Microsoft Word 2003 [152] and Adobe Reader 7 [5]. This section describes the architecture of AppMonitor.

The AppMonitor system consists of three parts: First, there is the main application program `AppMonitor.exe`, which is responsible for the majority of the system functionality. Second, there is the event-hooking DLL, `hooker.dll`, which is loaded into the memory space of the monitored applications. Third, there is a small secondary DLL, `MSWordStat.dll`, that has the sole purpose of determining the length of a document opened in Microsoft Word. Each of these sub-systems interact together and with the monitored applications.

4.3.1 Interaction with Monitored Applications

Figure 4.1 shows the high-level architecture of the AppMonitor system, with the arrows representing message passing between layers and applications. The main AppMonitor application runs on top of the Windows XP platform and is responsible for the majority of the functionality in the system, including displaying the real-time list of events, keeping track of open applications and documents, monitoring their state and coordinating the accompanying DLLs.

The hardware layer coordinates the transmission of input device signals (keyboard and mouse interrupts) to the operating system (Windows XP), which normally directs these messages to the appropriate application. However, AppMonitor places the event interception code of `hooker.dll` between the operating system and the monitored applications. `Hooker.dll` intercepts the mouse and keyboard events, records them if necessary, and passes them, unmodified, to the application. This is the method for obtaining low-level mouse events such as “left mouse button depressed” or “mouse moved” and low level keyboard events such as “Ctrl-f pressed”. These events are subsequently communicated to AppMonitor

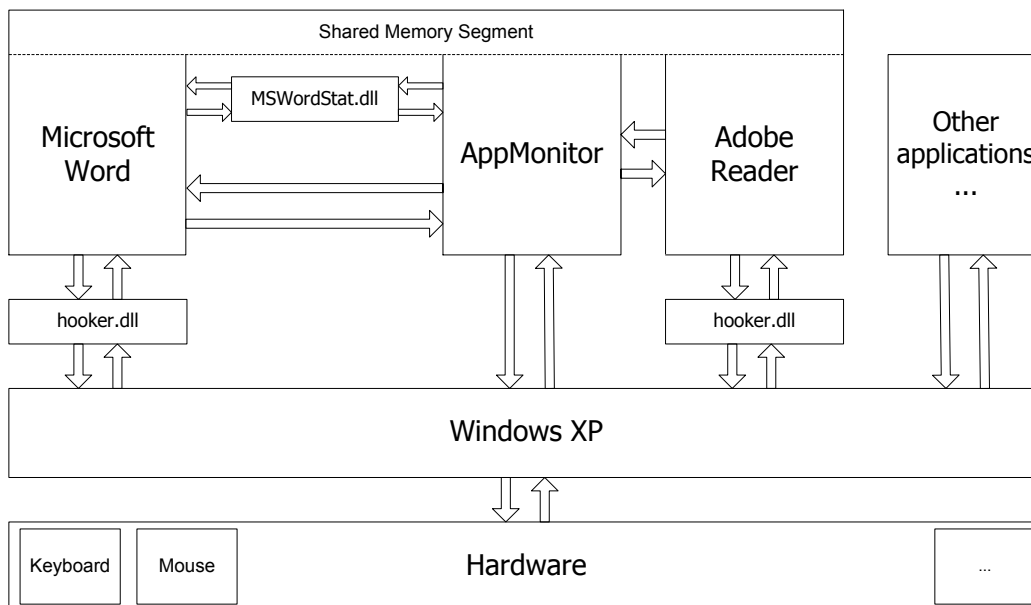


Figure 4.1: AppMonitor's high-level architecture

using a shared memory segment. The main AppMonitor executable also communicates with the applications directly using the Active Accessibility Interface (see Section 4.3.5). This communication allows AppMonitor to determine the interface semantics of the associated low-level event (for example, that a particular menu item is selected or the distance that the scrollbar is moved). AppMonitor also polls the monitored applications to determine whether the scrollbars or zoom of each document has changed. Polling is necessary because these interface controls change state not only through direct user manipulation, but also indirectly (and belatedly, because of threading) due to activities such as window-resizing, changes of view, or keystrokes to add or delete text. Finally, `MSWordStat.dll` is used to determine the length of Microsoft Word documents as they are opened.

Internally, AppMonitor must schedule polled events and handle the low-level interrupts as efficiently as possible. It must avoid significant delays in processing events to prevent skewed timings and more importantly, interrupting user interaction with the logged applications.

4.3.2 *Event Scheduling*

An outline of the internal scheduling of AppMonitor is shown in Figure 4.2. There are essentially two streams of events: those originating from regular polling (left hand column), and those arising from low-level interrupts (right hand column). An internal timer triggers every 200 ms, causing the interrogation of the list of known documents, the scrollbars of all open documents and other document properties. This interval is short enough to capture changes in scrollbar position or zoom level that can occur multiple times a second, but long enough to avoid performance degradation. Every second, the window hierarchy is traversed to determine the existence of any new documents—a higher resolution for traversal is not required as AppMonitor must wait until the window hierarchy is stable before logging (see Section 4.5.4). Finally, once every ten minutes, the local log file size and time since last sending are checked and if appropriate, the log file is uploaded. In practice it was determined the log file would rarely exceed this limit more often, unless full mouse and keyboard logging was enabled. The low-level interrupts arriving from the keyboard, mouse and WinEvents are processed as they arrive. The following sections detail further the use of the windows hierarchy to determine interface semantics within applications and how low-level events, high-level events and WinEvents are monitored.

4.3.3 *The Window Hierarchy*

The window hierarchy describes the internal structure of all windows and controls, with the desktop pane as the root. Every application has at least one top level window in the hierarchy, even when iconified. AppMonitor uses the window hierarchy to first identify when an application of interest is opened (either Microsoft Word or Adobe Reader) and then to identify pertinent controls within those windows for inspection (for example the scrollbar widgets).

AppMonitor uses the Windows Software Development Kit (SDK) function `EnumWindows` (Listing 4.1, line 4), in conjunction with a custom callback function (`checkWindow`, line 8) to test each top level window, to determine if it is a window of interest. Each window is tested with the function `GetClassName` (line 12) to determine its parent application. Microsoft Word windows are instances of the class `OpusApp` (line 15) and Adobe Reader windows are instances of the class

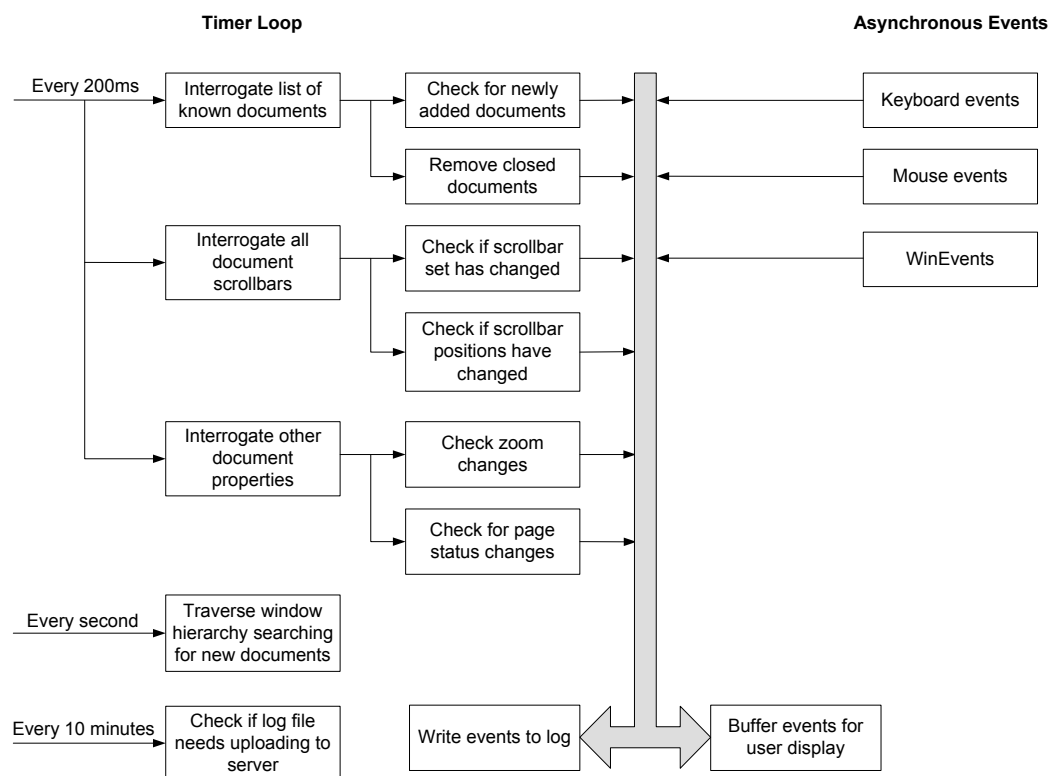


Figure 4.2: AppMonitor's internal scheduling architecture

```

1  /* Enumerate all top level windows, checking their application type with
2  the 'checkWindow' function */
3  ...
4  BOOL enumRes = EnumWindows(&checkWindow, (LPARAM) this);
5  ...
6
7  /* Check the application type of a window */
8  BOOL CALLBACK checkWindow(HWND hwnd, LPARAM lParam)
9  {
10     char name[MAX_NAME_LENGTH]; memset(name, 0, MAX_NAME_LENGTH);
11
12     if (GetClassName(hwnd, name, MAX_NAME_LENGTH) == 0)
13         return TRUE;
14
15     if (strncmp(name, "OpusApp", 7) == 0) {
16         checkAndProcessWordWindow(hwnd);
17     } else if (strncmp(name, "AdobeAcrobat", 12) == 0) {
18         checkAndProcessAdobeReaderWindow(hwnd);
19     }
20     return TRUE;
21 }

```

Listing 4.1: Top level window iteration to identify applications of interest (simplified for presentation)

AdobeAcrobat (line 17). In the current version of AppMonitor, only windows belonging to Microsoft Word or Adobe Reader are further scrutinised—extension to other applications requires modification of this filtering code.

Two tools greatly assist programmers in comprehending the internal window structure of applications: *Spy++* [162] and *Accessibility Explorer* [163]. Microsoft Windows based GUIs have one high level window that contains a series of ‘child’ windows to create the interface seen by the user. *Spy++* allows the developer to view the window handles,¹ class names and descriptions of all open windows and their children. It also displays system generated messages, such as mouse clicks, movements and key presses. An example of the *Spy++* display is shown in Figure 4.3. The figure illustrates part of the internal window structure of Microsoft Word, with the window handle, title and class indicated. Once AppMonitor identifies that the top level window belongs to Microsoft Word, it can traverse the hierarchy to identify the two scrollbar windows (also labeled). Sev-

¹A unique identifier [168].

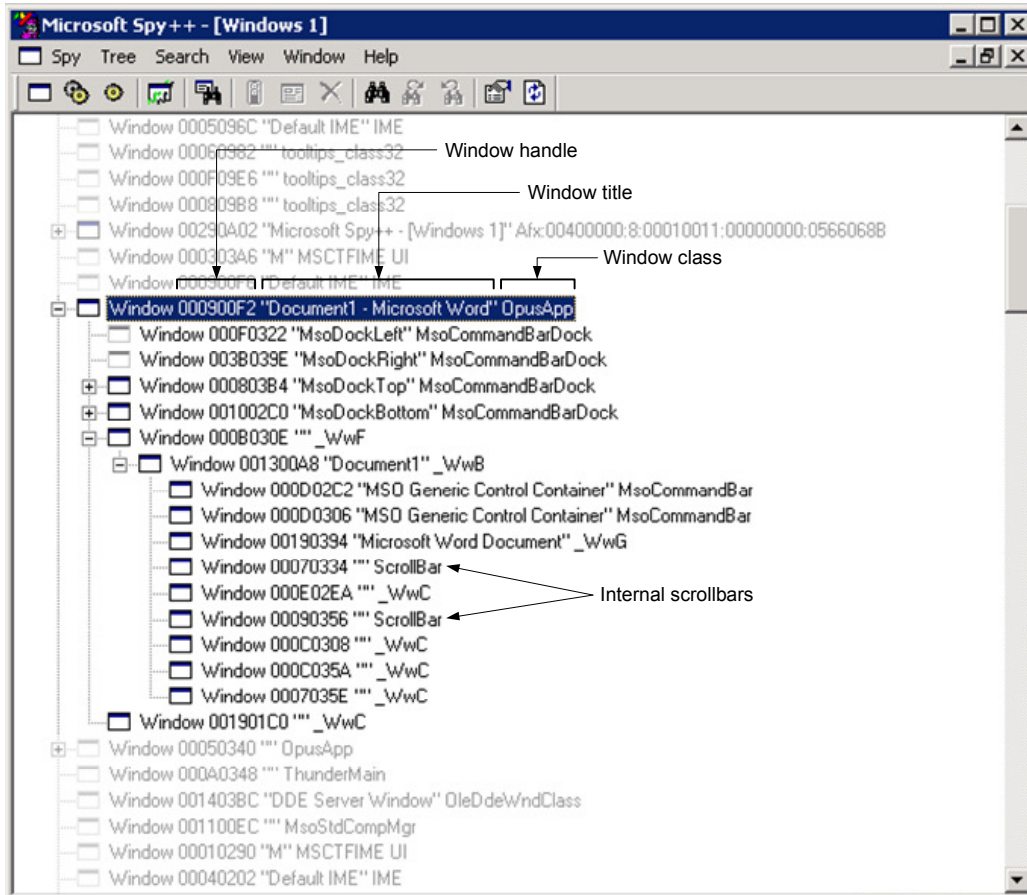


Figure 4.3: Spy++ displaying part of the window hierarchy. The windows not relevant to the Microsoft Word document are greyed for clarity.

eral similar tools exist to aid Microsoft .NET programmers using Windows Forms, for example ManagedSpy [234].

Accessibility Explorer allows programmers to view the *Accessible Object Tree*, which gives important additional details of the constituent components of an application, beyond that available with Spy++. For example, Spy++ describes a scrollbar as a single window, but Accessibility Explorer additionally discriminates between the thumb, trough and arrow components. This is especially useful when querying particular interface widgets for their value (see Section 4.3.6).

4.3.4 Low-level Event Logging

Once AppMonitor has identified an application of interest, it can use the window handle information to initiate low-level event logging. Microsoft Windows applications are event driven, meaning that active components await input to be passed to them via a message queue [164]. AppMonitor intercepts low-level events using the Windows Hook mechanism [143], which allows a program to be notified whenever another application is about to receive a message via its message queue. A ‘hook’ is installed by using the `SetWindowsHookEx` function, passing it the address of a callback function (to be called when a message is received). The hook callback function is responsible for passing the message onto the intended application. AppMonitor passes all messages on, unmodified, to the intended application to ensure the application’s behaviour does not change. AppMonitor’s hook procedures are stored in a DLL (`hooker.dll`), which is loaded into the address space of the logged application (Word or Reader). Keyboard and mouse events are both recorded as described below.

Key Events

AppMonitor can record all key presses, including regular typing events and keyboard shortcut commands. However, during longitudinal studies, participants are likely to be concerned that recording every key-press could allow the experimenter to recreate the document. Consequently, AppMonitor implements a key-filter that only passes key combinations that include a modifier key (either `Ctrl` or `Alt`), the arrow keys, the function keys, the navigation keys (page up, page down, home, end), enter or tab on to be logged. The remainder, the alphabetic and numeric keys, are passed without being logged, to the application.

Key presses are recorded in the log files using their virtual key code (for example, Listing 4.6, line 34). Key combinations that include a modifier are logically ORed with the modifier’s key code, as described in Table 4.1. Note that the `Shift` key by itself is not considered to be a modifier. However, it is useful to record if it is depressed when a modifier key is used. For example, `Ctrl` `→` will advance the cursor one word at a time, while `Ctrl` `Shift` `→` will advance the cursor one word at a time, while highlighting the text. A full list of virtual key codes is available from Microsoft’s website [165].

Modifier	Logical Mask
Ctrl	0x100
Alt	0x200
Shift	0x400

Table 4.1: Keyboard modifier masks

```

1  /* Add WordKeyboardProc as a keyboard 'hook' */
2  SetWindowsHookEx(WH_KEYBOARD, WordKeyboardProc, hInst, wordThreadID);

```

Listing 4.2: Initialising a keyboard hook for Microsoft Word

An example of key-event logging initialisation (‘installing the hook’) is shown in Listing 4.2. The function `WordKeyboardProc` is now called every time a key event occurs. A simplified version of `WordKeyboardProc` is shown in Listing 4.3. The function first determines which of the modifier keys are depressed (lines 5–12), checks whether this key should be logged using the `keyToSuppress` function (line 15) and if so, queues the event for logging (line 23). Finally, the keyboard event is passed, unmodified, onto the application (line 28). Listing 4.4 shows the process of determining which keys should and should not be logged.

Mouse Events

The mouse hooking mechanism can discriminate between all low-level mouse actions: movement, button depressions and releases, double clicks, and mousewheel use. Mouse ‘hooking’ is initiated similarly to keyboard hooking, as described in the previous section. The constant `WH_KEYBOARD` in Listing 4.2 is replaced with `WH_MOUSE` and an appropriate callback function address passed. `AppMonitor`’s callback functions for mouse events are more complex than those for the keyboard. Mouse presses and releases record the interface object under the cursor at the time of the event. Mouse movement events also record this information, as well as the position of the cursor on-screen. The objects under the cursor during a Mousewheel event are irrelevant, however `AppMonitor` records whether the `Ctrl` key is down during rotation of the mousewheel—the `Ctrl`-Mousewheel combina-

```

1  /* Callback function for Microsoft Word's keyboard hook */
2  LRESULT CALLBACK WordKeyboardProc(int nCode, WPARAM wParam, LPARAM lParam)
3  {
4      /* Determine which modifier keys are depressed */
5      SHORT ctrl = GetAsyncKeyState(VK_CONTROL);
6      SHORT alt  = GetAsyncKeyState(VK_MENU);
7      SHORT shift = GetAsyncKeyState(VK_SHIFT);
8
9      /* Short is 16 bits, if MSB is set, key is down */
10     ctrlDown = ((ctrl & 0x8000) == 0x8000);
11     altDown  = ((alt  & 0x8000) == 0x8000);
12     shiftDown = ((shift & 0x8000) == 0x8000);
13
14     /* Log the key event, if appropriate */
15     if (!keyToSuppress(wParam) || ctrlDown || altDown) {
16         int code = keyDown ? WM_KEYDOWN : WM_KEYUP;
17         if (isLogged(code)) {
18             WPARAM key = wParam | (ctrlDown ? CTRL_MASK : 0)
19                             | (altDown  ? ALT_MASK  : 0)
20                             | (shiftDown ? SHIFT_MASK : 0);
21             wchar_t keyBuff[20];
22             _snwprintf_s(keyBuff, 20, _TRUNCATE, L"0x%x", key);
23             queueEventRecord(code, NULL, keyBuff, NULL, 0, 0);
24         }
25     }
26 }
27 /* Pass the unmodified key event to the next hook in the chain */
28 return CallNextHookEx(getWordKeyboardHook(), nCode, wParam, lParam);
29 }

```

Listing 4.3: Keyboard callback function for Microsoft Word (simplified for presentation). keyDown on line 16 is defined in the omitted code.

```

1  /* Determine whether to ignore this key */
2  bool keyToSuppress (WPARAM wParam) {
3      return
4          (wParam >= '0' && wParam <= '9') ||
5          (wParam >= 'A' && wParam <= 'Z') ||
6          (wParam >= 'a' && wParam <= 'z') ||
7          (wParam >= VK_NUMPAD0 && wParam <= VK_DIVIDE) ||
8          (wParam > VK_RMENU && wParam < VK_ATTN) || // OEM keycodes
9          wParam == VK_BACK || wParam == VK_SPACE ||
10         wParam == VK_INSERT || wParam == VK_DELETE;
11 }

```

Listing 4.4: Function that determines which keycodes to ignore (not log)

tion triggers a zoom action in both of the monitored applications.

AppMonitor’s mouse and keyboard logging may not correctly output these events if the user installs software that translates the raw input from the hardware devices before passing it to the application. Software of this nature is useful for translating a QWERTY keyboard to Dvorak keyboard or to reprogram mouse button functionality. AppMonitor’s output would depend on the position in the event chain of this software.

4.3.5 High-level Event Logging—WinEvents

High-level event logging involves recording *WinEvents* that are generated by an application when its logical state changes. This allows AppMonitor to record some of the semantics of interaction that cannot be inferred from low level logs, such as “scrolling starting” or “menu item selected”. This is made possible by exploiting the *WinEvent Hook* functionality, which is part of the *Active Accessibility* Application Programming Interface (API).

Active Accessibility [166] is an API within the Windows SDK that is designed to ease the construction of interfaces for people with vision, hearing or motion impairments. For example, a programmer might use the Active Accessibility API to write a tool that allows a quadriplegic person to activate a menu using voice commands. To hook WinEvents, a program calls the `SetWinEventHook` API function, passing the location of an associated callback function to be run when the event occurs. Like the mouse and keyboard hooks, the WinEvent hook procedure must also be placed inside a DLL, in AppMonitor’s case `hooker.dll`.

4.3.6 Polling

Much of the interface state information such as the scrollbar position and current document zoom can only be determined by directly querying specific window objects inside an application. AppMonitor employs a polling mechanism to continually query each document in each application for any changes in state. Polling is necessary because certain interface components, such as scrollbars, can be updated without direct user intervention; for example, as a side-effect of changing the zoom-state. AppMonitor implements a polling interval of 200 ms, which is a trade-off between increased CPU demands at short intervals and failure to register

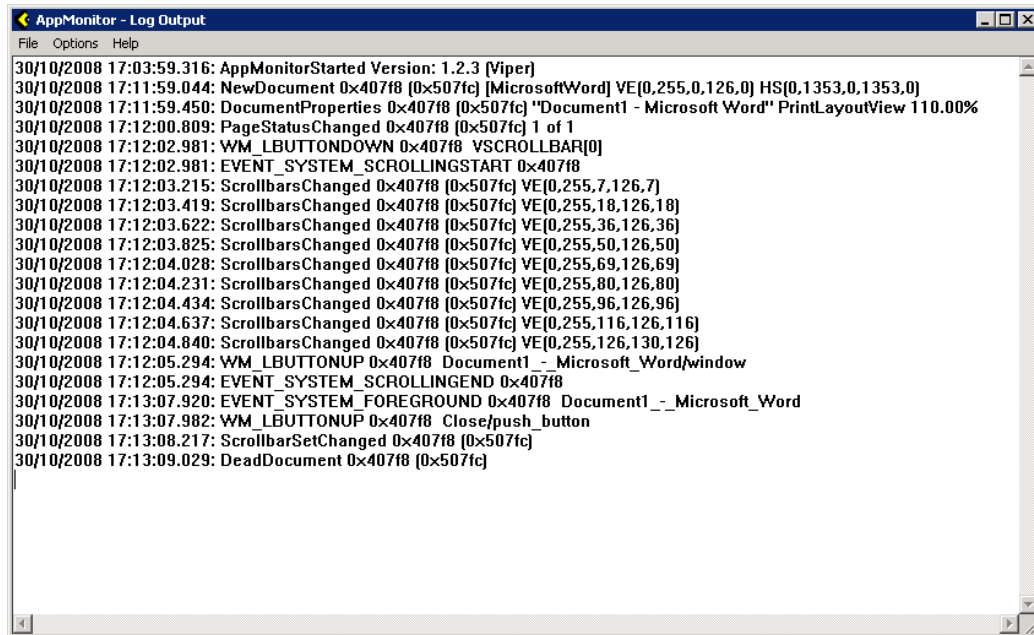


Figure 4.4: AppMonitor's event display window

pertinent events with longer ones. No issues of event loss were encountered with this polling interval.

4.3.7 User Interface

Under normal running circumstances, users are not required to interact with AppMonitor. The only indication that it is running is the small icon in the Windows system tray (🐍). Windows will hide "inactive" icons and so even this indicator may not be visible. The most important part of the user interface is the display of the events as they are recorded. This allows users to visually tracked logged actions (see Figure 4.4). The log display window appears when the user double-clicks the system-tray icon. The most recent event is shown at the bottom of the screen (with the scrollbar automatically moving to the bottom). The window shows the last 300 lines recorded in the log file. AppMonitor's logging can be stopped by selecting *File* → *Exit* from the menus associated with the log display.

4.4 Portability and Extendability

AppMonitor was built to observe interaction with Microsoft Word 2003 and Adobe Reader 7 when running under Microsoft Windows XP. This section provides guidance for researchers who wish to update or extend AppMonitor to newer software versions and to new applications.

AppMonitor is portable to any Microsoft Windows operating system that implements the technologies described in the *System Architecture* section (for example, Microsoft Windows Vista). It can also be extended to monitor any Microsoft Windows application that implements the Microsoft Active Accessibility Interface (this can be validated by the correct operation of Accessibility Explorer when querying the interface). Doing so requires an understanding of, and ability to modify, the following aspects of AppMonitor's operation. AppMonitor recognises when applications of interest are opened by regularly traversing the window hierarchy, comparing the root application class names with those that are to be logged. Once an application of interest is 'discovered' the window handle is used in the construction of an internal model for that application. To extend AppMonitor to a new application, researchers need to determine the internal application class name and add custom code for model instantiation.

Microsoft Word and Adobe Reader required a finer-grained model—one at document level rather than application level. This therefore required AppMonitor to continually inspect the application's internal window hierarchy to determine whether new documents were opened. Monitoring a new application will require the researcher to determine the granularity of model required, implement the model (see below) and write the application specific code to allow these models to be instantiated at the correct time.

The internal models maintain window handle references and state information about the application or document under scrutiny. The model must maintain a reference to the root window handle of the application to ensure it is only recognised as a new instance once. The model should also maintain state information and function implementations for any application specific monitoring that is to occur. Generally these functions will be called as part of the polling process, inspecting the internal state of the application, possibly through external DLLs.

Each application may also need custom keyboard or mouse 'hooks' to be

written inside `hooker.dll` if additional computation or extended logging capabilities are required. For example, when the mousewheel is moved, the current callback determines whether the `Ctrl` key is depressed, and if so, records `CtrlScrollWheel` (instead of simply `ScrollWheel`), as this is an important method of zoom control in document navigation systems.

The AppMonitor system uses the Active Accessibility Interface to record interaction with interface widgets such as menus, buttons and dialog boxes. These should not need to be modified to record actions in other applications.

Almost all of the events that may be of interest to researchers are exposed by the low-level, high-level, WinEvent or polling data capture techniques. However, sometimes a researcher is interested in system information that is not available through these channels. In such situations, a programmer may be able to find other means of determining the required information. One example of this is the small C# DLL that uses the Microsoft Office Interoperability API to record a Word document's length (see Section 4.5.2).

4.5 Implementation Issues and Lessons Learned

Like any software development project, several implementation issues were encountered during development of the AppMonitor logging tool. These issues are recorded below to help researchers wishing to develop similar systems or extend the AppMonitor application.

4.5.1 Defensive Programming Techniques

Defensive programming (also known as *secure coding*) is a broad term that encompasses best practises for computer programmers, ranging from SQL injection to the secure storage of data. Two of these techniques are especially important for AppMonitor: buffer overrun protection and robust error handling.

AppMonitor regularly queries the monitored applications for information regarding the state, role and value of manipulated objects. These values are usually short and descriptive (such as `View/menu_item`, Listing 4.6, line 17). However, when an interface widget's value changes according to user input, the length of the returned value could exceed the size of the buffer available. If left unchecked, this can induce application crashes, not only of AppMonitor, but potentially also

```
1 _snprintf_s(pEvent->nameObject, OBJECT_NAME_LENGTH, _TRUNCATE, "%S", objectName);
```

Listing 4.5: Defensive programming using a secured printf function

the monitored application (as `hooker.dll` is loaded into the memory space of Word and Reader). To prevent this occurring, all string manipulation is performed using ‘secured’ functions that require and check buffer sizes during copying, as shown in Listing 4.5.

Application loggers, like AppMonitor, must also robustly handle function call errors and failures. Monitoring external applications makes this issue more likely—for example, AppMonitor has no control over when windows are closed; however, much information retrieval relies on the presence of a top-level or internal window. The AppMonitor code checks for errors and makes provision for failed function calls throughout.

4.5.2 Accessing a Microsoft Word Document’s Page Length

The length of a document allows the internal position and length of the scrollbar to be converted into a page value. Both Microsoft Word and Adobe Reader display the length of the current document on their interface. Adobe Reader’s page counter is a child-window of the application, with its contents accessible using the `GetWindowText` function. Word’s page counter is inside a status bar widget that does not provide such convenient access.

The only practical method of acquiring the length of a Word document is by using the Microsoft Office Interoperability API. AppMonitor implements this functionality in a C#/ .NET DLL, `MSWordStat.dll`. While the content of this DLL is relatively straightforward, adding interoperability between the primary C/C++ application and the new DLL increased the complexity of AppMonitor. It also increased the requirements on the user’s computer—the .NET runtime must also be installed (this turned out to not be a problem, as this is already installed on most Windows computers).

Finally, asking for the length of a Word document, in pages, causes the document to be repaginated. For a small document on a fast machine the user is unlikely to notice this take place. However, on slower machines and especially in

large documents, the user will observe Word processing the document as it repaginates. For this reason, the page length of a document is only queried once, as the document is opened. This means that user editing actions that increase the page length of a Word document will not explicitly be recorded. However, an increase in the scrollbar's maximum value and a decrease in the thumb's extent, without an associated action (such as zooming) can be correlated to a change in length.

4.5.3 Using the Active Accessibility Interfaces with Adobe Reader

The Active Accessibility Interface API is designed to allow programmers to create applications such as screen readers for the visually-impaired. It does this by providing access to the underlying structure of the interface. The Adobe Reader software detects when this API is accessed and opens a dialog asking the reader to specify the reading order of an untagged document. Usually, this is important, so that applications such as screen readers can have the document content provided in the correct order (for instance in a two column document the text is read down one column and then the next, rather than line-by-line across the complete page width). Unfortunately, this dialog box also appears when AppMonitor queried the Adobe Reader interface for state information.

The second system requirements goal stipulated that users should be allowed to continue with their everyday work in unmodified applications. Asking participants to manually close this dialog would mean this goal could not be achieved. One option was to remove the accessibility plug-in from Adobe Reader's plug-in directory. However, this would mean modifying the application, possibly causing other unexpected side-effects, and again conflicting with the second requirements goal. It was determined that this dialog could be prevented by modifying three registry keys. The key names and the required values are shown in Table 4.2, to aid other researchers who may also encounter this issue.

4.5.4 Delaying Logging until an Application's Window Structure is Stable

AppMonitor detects new documents when the top level window of the application appears in the window hierarchy. However, the presence of a window in this hierarchy does not reflect the progress of the system creating and loading the application for use. During loading, applications are unstable, and attempting to

Registry key	Value
bCheckReadMode	0x00000000
iReadingMode	0x00000001
iWizardRun	0x00000001

Table 4.2: Registry key values used to prevent Adobe Reader opening a *reading order* dialog for untagged documents. To be used in conjunction with the prefix HKEY_CURRENT_USER\Software\Adobe\Acrobat_Reader\7.0\Accessibility\

traverse the internal structure of the application during this period can result in application crashes.

AppMonitor implements a time-delay mechanism, by counting WinEvents from the application, to avoid traversing the hierarchy too early. It was experimentally determined that after receiving approximately 50 WinEvents (caused by the construction of the user interface), an application’s window hierarchy would be stable. At this point, information regarding the state of the page layout, zoom and scrollbar positions can be safely logged. This threshold is likely to be application dependent.

4.5.5 Unimplemented Functions

AppMonitor’s mouse callback implements generic functionality to query the object under the mouse cursor, when button presses and releases are detected. The callbacks use Accessibility API functions such as `get_accValue` to obtain the value and `get_accRole` to obtain the role of the underlying interface object. Unfortunately, calling these methods of particular objects in Word and Reader cause the application to inexplicably crash.

In Microsoft Word, calling the `get_accValue` method on some Listbox objects causes this problem. The same issue arises in Adobe Reader when the `get_accRole` method is called on the zoom drop-down menu. To resolve these issues, custom code was added to the mouse callback functions in order to check for these objects before querying for state information. These issues came to light during the beta-testing period of AppMonitor.

There are two lessons other researchers can learn from AppMonitor encountering these issues. First, developers should not completely trust that all interface widgets will respond correctly to requests for state information through the Active Accessibility APIs. Thorough testing of the application under investigation should always be conducted. Second, this example demonstrates the benefits of having a small number of beta-testers use the logging software before it is widely distributed.

4.6 Configuration

AppMonitor provides researchers with a configuration GUI for tailoring the set of events to be logged when the software is distributed to participants. When AppMonitor is run in Visual Studio's *debug* mode a configuration menu item is enabled that displays the dialog shown in Figure 4.5. The interface provides mechanisms for adding and removing events to be logged. Removing uninteresting events can reduce both the size of the log files and the load on the host machine. The configuration changes are saved to a file that can be packaged and distributed with the system. To ensure a consistent set of data is collected across participants, the configuration interface is disabled when the system is compiled in *release* mode, as would be the case when AppMonitor is distributed to study participants.

4.7 Log Files

All events from a particular user are stored in a single log file on the local machine. This file is automatically uploaded to a remote web server whenever it reaches a threshold size or after a week has passed since the last upload. A CGI script on the server receives the log file and then instructs the host machine to truncate the file it holds. This method of file transfer occurs over the HTTP port 80 (as used for Internet browsing) and so prevents the need for system administrators to allow traffic on other ports before AppMonitor can correctly function. Listing 4.6 shows a sample log file produced by AppMonitor when Microsoft Word is in use. Line numbers have been added for clarity of explanation. The log file demonstrates many of the capabilities of AppMonitor, as described below.

```

1 17/08/2006 14:21:42.265: NewDocument 0x30029c (0x2f0274) [MicrosoftWord] VE(0,255,0,62,0) HS(0,1355,0,1355,0)
2 17/08/2006 14:21:43.437: DocumentProperties 0x30029c (0x2f0274) "AppMonitor.doc - Microsoft Word" PrintLayoutView 150.00%
3 17/08/2006 14:21:50.657: PageStatusChanged 0x30029c (0x2f0274) 1 of 5
4 17/08/2006 14:22:23.805: WM_LBUTTONDOWN 0x30029c VSCROLLBAR[0]
5 17/08/2006 14:22:23.805: EVENT_SYSTEM_SCROLLINGSTART 0x30029c
6 17/08/2006 14:22:24.366: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,2,15,0)
7 17/08/2006 14:22:24.766: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,6,15,0)
8 17/08/2006 14:22:25.167: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,11,15,0)
9 17/08/2006 14:22:25.247: WM_LBUTTONUP 0x30029c grip
10 17/08/2006 14:22:25.247: EVENT_OBJECT_STATECHANGE 0x30029c Line_down
11 17/08/2006 14:22:25.247: EVENT_SYSTEM_SCROLLINGEND 0x30029c
12 17/08/2006 14:22:25.367: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,11,15,11)
13 17/08/2006 14:22:32.367: WM_MOUSEWHEEL 0x30029c Scrollwheel
14 17/08/2006 14:22:32.377: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,13,15,13)
15 17/08/2006 14:22:32.708: WM_MOUSEWHEEL 0x30029c Scrollwheel
16 17/08/2006 14:22:33.178: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,16,15,16)
17 17/08/2006 14:22:41.140: WM_LBUTTONDOWN 0x30029c View/menu_item
18 17/08/2006 14:22:41.140: EVENT_SYSTEM_MENUSTART 0x30029c AppMonitor.doc--_Microsoft_Word/Menu_Bar
19 17/08/2006 14:22:41.150: EVENT_SYSTEM_MENUPOPUPSTART 0x30029c View
20 17/08/2006 14:22:41.240: WM_LBUTTONUP 0x30029c View/menu_item
21 17/08/2006 14:22:44.835: WM_LBUTTONDOWN 0x30029c Thumbnails/menu_item
22 17/08/2006 14:22:44.975: WM_LBUTTONUP 0x30029c Thumbnails/menu_item
23 17/08/2006 14:22:44.975: EVENT_SYSTEM_MENUPOPUPEND 0x30029c View
24 17/08/2006 14:22:44.995: EVENT_SYSTEM_MENUEND 0x30029c AppMonitor.doc--_Microsoft_Word/Menu_Bar
25 17/08/2006 14:22:45.116: ScrollbarsChanged 0x30029c (0x2f0274) HS(0,1284,114,634, 114)
26 17/08/2006 14:22:45.196: ScrollbarSetChanged 0x30029c (0x2f0274) VE(0,255,16,15,16) HS(0,1284,114,634,114) (0,255,0,224,0)
27 17/08/2006 14:22:45.396: ScrollbarsChanged 0x30029c (0x2f0274) VC(0,255,0,224,0)
28 17/08/2006 14:23:09.100: WM_LBUTTONDOWN 0x30029c Zoom:/combo_box[150%]
29 17/08/2006 14:23:09.100: EVENT_SYSTEM_MENUSTART 0x30029c AppMonitor.doc--_Microsoft_Word/Menu_Bar
30 17/08/2006 14:23:10.632: WM_LBUTTONUP 0x30029c 100%/list_item
31 17/08/2006 14:23:10.652: EVENT_SYSTEM_MENUEND 0x30029c AppMonitor.doc--_Microsoft_Word/Menu_Bar
32 17/08/2006 14:23:10.833: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,0,22,0) HS(0,856,68,634,68)
33 17/08/2006 14:23:10.833: ZoomChanged 0x30029c (0x2f0274) 100.00%
34 17/08/2006 14:23:38.422: WM_KEYDOWN 0x30029c 0x143
35 17/08/2006 14:23:38.593: WM_KEYUP 0x30029c 0x143
36 17/08/2006 14:23:42.779: WM_LBUTTONDOWN 0x30029c VSCROLLBAR[0]
37 17/08/2006 14:23:42.789: EVENT_SYSTEM_SCROLLINGSTART 0x30029c
38 17/08/2006 14:23:43.279: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,2,22,2)
39 17/08/2006 14:23:43.680: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,73,22,73)
40 17/08/2006 14:23:44.080: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,146,22,146)
41 17/08/2006 14:23:44.281: ScrollbarsChanged 0x30029c (0x2f0274) VE(0,255,157,22,157)
42 17/08/2006 14:23:44.491: WM_LBUTTONUP 0x30029c VSCROLLBAR[61]
43 17/08/2006 14:23:44.501: EVENT_SYSTEM_SCROLLINGEND 0x30029c
44 17/08/2006 14:23:46.945: WM_LBUTTONDOWN 0x30029c Microsoft_Word_Document/ client
45 17/08/2006 14:23:47.075: WM_LBUTTONUP 0x30029c Microsoft_Word_Document/client
46 17/08/2006 14:23:49.909: WM_KEYDOWN 0x30029c 0x156
47 17/08/2006 14:23:50.079: WM_KEYUP 0x30029c 0x156
48 17/08/2006 14:24:06.864: WM_LBUTTONDOWN 0x30029c Close/push_button
49 17/08/2006 14:24:07.805: WM_LBUTTONUP 0x30029c Close/push_button
50 17/08/2006 14:24:07.915: ScrollbarSetChanged 0x30029c (0x2f0274)

```

Listing 4.6: Sample AppMonitor log file, for a Microsoft Word session

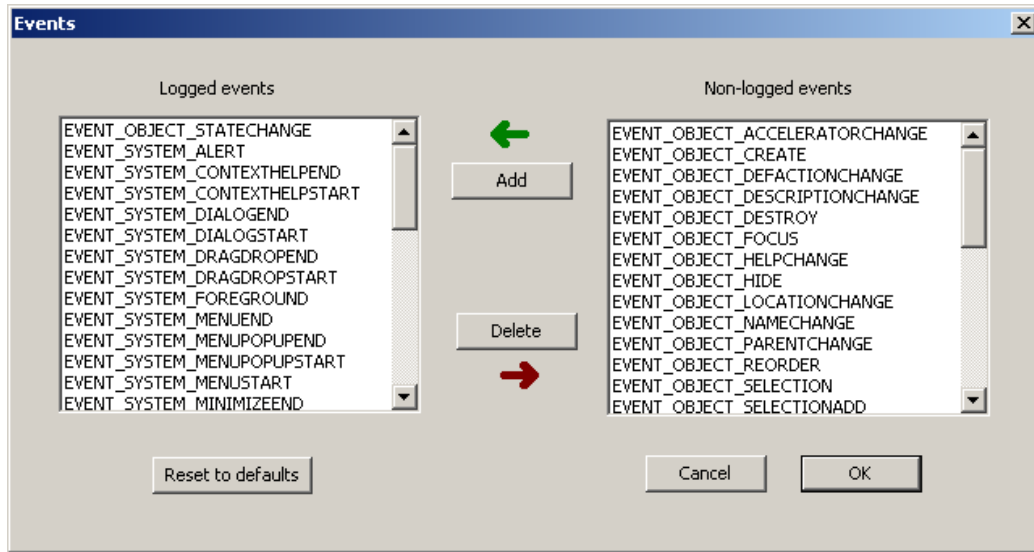


Figure 4.5: AppMonitor's configuration dialog

Every line in the log file begins with a date and timestamp (down to millisecond accuracy²), followed by an event code, one or more identifying window handles and possibly further information about the event. There are two types of event codes:

Operating system event codes: These are fully capitalised in the log file and are generated by the Windows SDK (for example, Listing 4.6, line 4, WM_LBUTTONDOWN). These events originate from the keyboard, mouse and WinEvents.

AppMonitor pseudo-event codes: These are the mixed case event codes and are generated by AppMonitor (for example, Listing 4.6, line 1, NewDocument). These events originate from all other data collection techniques, for example, polling.

The window handles (hexadecimal numbers, for example, Listing 4.6, line 1, 0x30029c) uniquely identify the document and application that has generated the

²AppMonitor timestamps all registered events. However, the operating system may introduce a slight delay between an event occurring and AppMonitor receiving notification (for example, under high-load conditions). Hence, millisecond timestamp accuracy cannot be guaranteed.

event. These allow AppMonitor to distinguish between documents that are open concurrently in both the same application and in a different application.

Line 1 shows that AppMonitor has detected a document being opened in Microsoft Word. The code `VE(0,255,0,62,0)` describes the *Vertical East* scrollbar. The numerical values in the brackets describe the current state of the scrollbar, using the convention: (minimum trough value, maximum trough value, scrollbar static position, document thumb size, scrollbar dynamic position). The static scroll position is not updated when the user drags the scroll thumb; however, the dynamic position is updated (allowing the position of the scroll thumb to be determined even if scrolling is under way). In this case, the scroll trough extends from 0 (the top) to 255 (the bottom) with the static position of the thumb currently 0. The thumb size is $62/255^{\text{ths}}$ of the gutter length. Finally, it has a dynamic scroll position of 0.

The second line describes the state of the document at the time of opening. The document `AppMonitor.doc` is opened in Microsoft Word's *Print Layout* view, with an initial zoom of 150%. Line 3 shows that the document is five pages long.

Lines 5–11 show the user scrolling from the top of the document to a position 5% of the way through the document (line 8). The fact that the button comes up on the *grip* (line 9) having used the *line down* control (line 10) shows that the scrollbar's down arrow was used. Lines 12–16 record mousewheel scrolling actions. Each of the `ScrollbarsChanged` events record the position of the scrollbar at a particular point in time, allowing post-processing to determine the speed and direction(s) of the scroll action.

Lines 17–20 show the user posting the *View* menu and then selecting the *Thumbnails* menu item (lines 21–22). Lines 25–27 show scrollbar updates caused by displaying the thumbnail panel. This feedback can be discriminated from end-user scrolling because it is not accompanied by a scroll action.

Lines 28–33 show the user changing the zoom level from 150% to 100% using the zoom combo-box (line 28). On lines 34–47 the user carries out a copy/scroll/paste action using `Ctrl-c` on lines 34–35, the scroll thumb (lines 36–43) and `Ctrl-v` (lines 46–47). The hex-code for `Ctrl-c`, `0x143` constitutes the virtual key code for the 'c' character (`0x43`) logically 'ORed' with a mask representing the `Ctrl` modifier (`0x100`).

Finally, the user closes the application using the close button at the top right

corner of the window (lines 48–50).

4.8 Parsing and Analysing Log Files

AppMonitor’s log files are generated in a structured manner to aid readability, both for humans and machines. AppMonitor can potentially generate a large amount of data over time, meaning that human parsing and summarisation of actions is impractical. To aid this process, the AppMonitor log files follow a strict syntax definition. The BNF syntax for the log files, in its complete form, is described in Appendix C. Briefly, each line begins with a date and time, followed by the event name, one or more window handles to identify the application and document, and possibly extra event information. AppMonitor specific messages (such as informing of a file upload) are an exception to this rule. Parsing scripts for the studies reported in this thesis were written in the Python programming language [187].

4.8.1 Dividing Log Files into Documents

AppMonitor’s log files contain the events recorded on a user’s computer ordered by date and time. This means the events from multiple documents used simultaneously are intertwined in the log files. For the majority of analyses, it is useful to divide the raw log files into individual files for each document opened. To do this, each event’s window handle(s) is checked and matched to a particular document.

For Microsoft Word documents, the two window handles are unique (due to Word’s Single-Document Interface) and easily allow this separation. However, some dialog boxes, such as the *Open File* dialog, have their own set of unique window handles, that bear no relation to those of the primary Word window. These should either be ignored or contextual parsing performed to detect these instances (for example by matching the press of the open-file button to the resulting events).

Adobe Reader’s Multiple-Document Interface (MDI) makes document separation more difficult. Events that are produced as the result of AppMonitor polling the application are identified by the two window handles. All other events can only be identified by the primary window handle. Parsing must use contextual information (such as the prior and following polled events) to determine the source document for the remaining events.

```
1 newDocument      = re.compile (".*NewDocument.*")
2 documentProperties = re.compile (".*DocumentProperties.*")
3 scrollbarsChanged = re.compile (".*ScrollbarsChanged.*")
4 deadDocument     = re.compile (".*DeadDocument.*")
```

Listing 4.7: Example regular expressions

```
1 #!/usr/bin/python
2
3 import re, sys
4
5 scrollbarsChanged = re.compile (".*ScrollbarsChanged.*")
6 file = open("example.log", "r")
7
8 for line in file.readlines():
9     if scrollbarsChanged.match(line):
10         print line
11
12 file.close()
13
14 sys.exit(0)
```

Listing 4.8: Example script to extract ScrollbarChanged events

4.8.2 Regular Expression Library

A large portion of log parsing involves pattern matching to extract relevant events. Python supports regular expression matching through the `re` library; the efficiency of matching in large data sets can be increased by precompiling the regular expressions using the `compile` function.

Parsing of the AppMonitor log files made regular use of such pre-compiled expressions and so they are combined into a regular expression library. Listing 4.7 shows a small part of this library. These regular expressions can now be very easily used, for example, the code required to extract the `ScrollbarsChanged` events from a file is shown in Listing 4.8.

4.8.3 Decoding Log File Events

AppMonitor uses encoding schemes for the `ScrollbarChanged` and keyboard events, for log file brevity. To allow human understanding of these two types of events, the information must be decoded.

```

1 def getPositionPercent(scrollbarString):
2     scrollbarString = scrollbarString.replace("(", ",").replace(")", "")
3     components      = scrollbarString.split(",")
4
5     sMin            = float(components[1])
6     sMax            = float(components[2])
7     staticPos       = float(components[3])
8     thumbSize       = float(components[4])
9     dynamicPos      = float(components[5])
10
11     percent = (staticPos / (sMax - sMin - thumbSize + 1)) * 100
12
13     return percent

```

Listing 4.9: Python function to convert a ScrollbarsChanged event into a percent representing the distance through the document.

ScrollbarsChanged events are encoded using five numerical values: (minimum trough value, maximum trough value, scrollbar static position, document thumb size, scrollbar dynamic position), as described in Section 4.7. Percentages were found to be the most amenable measure for comparing scrollbar positions across multiple, different length documents. The Python function shown in Listing 4.9 takes an input ScrollbarsChanged line and returns an equivalent percentage value, indicating the distance from the beginning of the document.

Key-events require conversion from their hexadecimal representations into human-readable form. Section 4.7 detailed the make-up of a key's representation. The decoding process involves recording and masking out the modifier keys, checking a look-up table for 'non-character' keys (the function keys, space, home, insert and so on) and finally making a conversion to a character if the key has not already been identified.

4.9 Summary

AppMonitor allows researchers to log document navigation actions in unmodified Windows applications. It does this by using low-level mouse and keyboard 'hooking', WinEvents, the Active Accessibility APIs and the Microsoft Office Interoperability API. AppMonitor successfully met the six design requirements detailed at the start of this chapter: it logs user actions in Microsoft Word and

Adobe Reader, it does not interfere with the user's everyday work, it allows the logged events to be viewed and it provides a mechanism for stopping logging, if required. Despite the potential to perform 'spyware' type functions (logging mouse and keyboard actions) users experience no issues with firewalls or anti-virus programs blocking its functionality. In summary, AppMonitor allows researchers to conduct longitudinal studies of document navigation actions in real world applications, over long periods of time.

The primary limitation of AppMonitor is the lack of contextual information describing the actions recorded. AppMonitor has a limited knowledge of the content of the opened documents (only the length, in pages). Different document types, such as conference papers, instruction manuals and books may have diverse interaction styles. AppMonitor also has no knowledge of the tasks the user is performing. Reverse engineering the log files can allow some insight, such as Ctrl-f indicating the user is searching for a phrase, but the higher level tasks are likely to be indiscernible.

Despite this limitation, AppMonitor provides a valuable source of data on the actions users are performing in unmodified Windows applications. It performed reliably throughout the studies described in this thesis. At the conclusion of the 120 day longitudinal study (reported in the following chapter) several participants commented that they had 'forgotten' AppMonitor was still running on their machines. This provides a good validation of its reliability and unobtrusive logging of user actions.

Several research groups around the world are actively using AppMonitor for a variety of projects. Commercial organisations have also shown interest in employing AppMonitor's logging techniques. The source code for AppMonitor is available by contacting the author.

Chapter 5

How do Users Navigate in Documents? An Empirical Characterisation of Electronic Document Navigation

RESEARCHERS investigating techniques for improving electronic document navigation would benefit from a thorough empirical characterisation that describes how current systems are used in their everyday environment. Insights into the documents used, the tools employed and patterns of navigation would provide evidence and motivation for the development of new navigation techniques. The longitudinal study described in this chapter provides such a characterisation, establishing a foundation for the re-design of document navigation tools. It used AppMonitor to record the navigation actions of 14 participants, using Microsoft Word and Adobe Reader, over a 120 day period.

The data analysis from this study describes many aspects of electronic document navigation, including the documents used, the attributes of navigation tool use and tool-independent patterns of navigation. The most significant findings from the study are as follows: the mousewheel, scrollbar thumb and paging keys are used for the majority of navigation; many tools lauded for their efficiency, including bookmarks and search tools, show low utilisation; half of users' navigation actions can be generalised across applications, while the remaining half modify their tool selection based on the application and; within-document revisitation is a commonly performed action, yet the tools that support this task are rarely used.

This chapter has the following structure. To begin, it describes the study goals, participants and methodology. The presentation of results is prefaced by an introduction to the methods of analysis and reasons for the assumptions made. From here, the remainder of the chapter characterises user actions during the study. This

includes the following: the properties of the documents used, analysis of vertical document navigation, a breakdown of the use of individual tools, horizontal document navigation, the use of page layout and zoom tools, navigation patterns observed, the use of reviewing tools, document and window management activities, and finally, the use of menu items, toolbar buttons and keyboard shortcuts. It concludes with a discussion and summary of the results of the study.

5.1 Study Goals

The primary goal of this study is to empirically characterise the everyday document navigation actions of users, over a four month period. The resulting characterisation will inform interaction designers on the properties of the documents used, the extent of use of currently available tools, stereotypical categories of user and higher-level navigation patterns. To achieve this goal, AppMonitor is used to monitor the navigation actions of the volunteer participants. Participants are asked to continue with their everyday work in Microsoft Word and Adobe Reader and to not modify their actions in any manner. The widespread use of Word and Reader (see Section 4.2) make these applications ideal candidates for observation—there is a high likelihood of user familiarity and the results will generalise to a large audience. The diversity of actions that AppMonitor can record also facilitates collection and reporting of related data, such as button presses and menu selections.

5.2 Participants

The longitudinal study monitored the navigation actions of 14 volunteer participants over a period of 120 days. The participants were all Computer Science staff and postgraduate students, two of whom were female. All participants used a mouse that was fitted with a mousewheel (or in one case, a touchpad's mousewheel emulator) that was controlled with their right hand. Participants were experienced computer users, nine of whom spent over 40 hours/week using a computer. Further demographic information is provided in Table 5.1.

General			
<i>Gender</i>		<i>Age</i>	
Male	12	Mean	31 years
Female	2	Standard deviation	11 years
Input			
<i>Dominant hand</i>		<i>Mouse control hand</i>	
Right	12	Right	14
Left	2	Left	0
<i>Mouse equipped with mousewheel?</i>			
Yes	13		
On laptop trackpad	1		
Vocation			
<i>Occupation</i>		<i>Area</i>	
Student	10	Computer Science	14
Faculty	4		
Computer use			
<i>Approx. hours/week using a computer</i>		<i>% hours computer used for work</i>	
20–30 hours	1	Mean	78.2%
30–40 hours	4	Standard deviation	14.1%
40–50 hours	3		
50+ hours	6		
<i>Self-rating of computer skills</i>			
Beginner	0		
Intermediate	0		
Confident	0		
Advanced	1		
Expert	13		
Microsoft Word			
<i>Approximate use of Microsoft Word</i>		<i>Use Microsoft Word to ...[†]</i>	
Occasionally	6	Read documents	13
Occasionally–regularly	2	Edit or review documents	10
Regularly	3	Create documents	13
All of the time	3		
Adobe Reader			
<i>Approximate use of Adobe Reader</i>			
Occasionally	1		
Occasionally–regularly	2		
Regularly	6		
All of the time	5		

[†] Multiple selections were allowed

Table 5.1: Demographic information and survey results for the fourteen longitudinal study participants

5.3 Methodology

All participants were given a brief introduction to the study and signed appropriate consent forms before AppMonitor was installed on their computer. Participants were given a brief summary sheet explaining the objectives of the study, what their participation would entail and their rights as a study participant (reproduced in Appendix D.1). They then signed a consent form (Appendix D.2) before AppMonitor was installed on their machine. AppMonitor's full keyboard and mouse logging was disabled to reduce the CPU demands on the host computer, to decrease the size of the log files and to protect user privacy.

Following installation, participants were shown how to view, in real time, the logged events and how they could stop AppMonitor's logging. Participants provided basic demographic information (Appendix D.3), before being asked to continue with work as normal—they would not be contacted until the completion of the study.

At the conclusion of the four month period, participants were asked to manually upload their final log file and uninstall AppMonitor. Finally, an invitation was extended to take part in an interactive session to help understand the reasons for their actions. The outcomes from these sessions are reported in Chapter 6.

5.4 Data Analysis Parameters, Assumptions and Result Presentation

To support replication and comparison, the study data should be analysed and presented consistently, and have assumptions clearly stated. This section describes several important preliminaries and assumptions made during data analysis. It begins with *Document Usage Sessions*—a method for measuring document interaction time. It then discusses explicit navigation actions and the grouping of events. Finally, important points pertaining to the presentation of results are described. The results of the data analysis are documented in the remaining sections of this chapter.

5.4.1 Document Usage Sessions

An early observation during data analysis was that users regularly left their documents open for hours, overnight or even days without any interaction. A *Doc-*

ument Usage Session (DUS) was defined to discard this idle time and to avoid over-inflation of the users' interaction time. These DUS's are useful when applying time analysis to the data recorded by AppMonitor.

A DUS is a period of time where the user is interacting with their document, or more accurately, a period of time where AppMonitor has recorded at least one event originating from that document. A session begins when an event for a particular document is registered. It continues until a period of five minutes has passed without AppMonitor collecting any events. The document is then classified as *idle*, with the assumption that the user is not interacting with the document in any manner. Usage time then restarts when the next event is recorded. A DUS is terminated when the document is closed.

The DUS timeout of five minutes is a trade-off between prematurely terminating an interaction session and exaggerating usage time when a user immediately moves to another document or application. Five minutes allows for short periods of reading or typing (recall that keyboard logging was disabled) at the same document position, while also accounting for situations where the user switches to another application.

A DUS timeout of ten minutes was also trialed. As expected, longer interaction times were recorded, however large differences were not observed. Periods of less than five minutes were estimated as too short—a user could easily spend this amount of time reading the same part of the document; however, it is unlikely they would spend ten or more minutes attending to a single position in the document. Any “missed” interaction time during the five minutes (caused by longer periods of reading or typing) is balanced by the time where users move immediately away from the application, while the DUS continues.

5.4.2 Only Explicit Navigation Actions are Considered

Only *explicit* navigation actions are considered during data analysis. Microsoft Word will often automatically change the position of the document while editing actions are underway. For example, when typing or pasting content, the document is moved up, to prevent the cursor from running off the end of the screen. The document's position may also change when zoom actions are performed or the page layout is modified. The analysis of the logs ignores automatic navigation

actions such as these and focuses on actions explicitly made by the user (such as dragging the scrollbar thumb).

5.4.3 *Grouping Navigation Events into Navigation Actions*

A timeout period of two seconds is used to group *navigation events* (single AppMonitor log file lines) from the same tool into *navigation actions*. This timeout means that *unintentional* pauses, for example, when re-clutching the mouse, do not split events that are part of the same action. To ease analysis, multi-directional navigation actions are also split into their uni-directional components.

AppMonitor's raw navigation events may constitute part or all of the use of a navigation tool. For example, every turn of the mousewheel generates a line in the log file; however, users rarely move a single 'click' at a time. Conversely, pressing the *next page* button generates two events (*mouse down* and *mouse up*), providing temporally well-defined beginning- and end-points for the action. The mousewheel example has no such beginning and end and instead requires a timeout period in order to group the generated events.

By reviewing the log files, trialing several timeout periods and informally testing the use of various navigation tools, it was concluded that events that could be grouped without a break of two seconds constituted a single navigation action. Shorter (one second) and longer (three second) timeout periods were also tried. Significant differences in the results were not observed; however, it was estimated that a one second timeout was too short, while a three second timeout was too long to correctly group the intended actions of the user.

For this analysis, navigation actions are also split into their uni-directional components. Some navigation tools, such as the scrollbar thumb and Adobe Reader's hand-tool, allow the user to change the direction of navigation without initiating a new action. In situations where this occurs, the navigation events are divided at the point of direction change. For example, quickly dragging the scrollbar thumb from the start of the document to the end and returning to the beginning, without releasing or pausing, would constitute two navigation actions—one for moving to the end of the document and one for returning to the top. This additional condition significantly eases analysis and result presentation by grouping actions into their simplest constitute form.

To summarise, the term *navigation action* is used to refer to a group of events from the same tool, without a break of more than two seconds, all moving in the same direction.

5.4.4 Presentation of Results

This characterisation distills large amounts of data into useful, meaningful summaries. For clarity and to avoid repetition, some important preliminaries for the presentation of the study results are discussed in the following paragraphs.

Data Aggregation and Calculation of Mean Values

When summarising data from many users, it is important to not let one user's actions be lost or overshadow those of the other participants. This can occur when one participant is at the extremities of the analysis scale, for example, if one user scrolled an average of one page in each document and the next user an average of 100 pages. Aggregating data can cause this effect.

To help prevent this issue, statistics are generally reported as the *mean of participant means* (MoPM). This value is calculated by first determining the mean value for each participant and then averaging those means to calculate a value for reporting:

$$\bar{x} = \frac{1}{n_p} \sum_{i=1}^{n_p} \left(\frac{1}{n_{vi}} \sum_{j=1}^{n_{vi}} v_j \right) \quad (5.1)$$

where:

- \bar{x} = reported mean
- n_p = number of participants
- n_{vi} = number of values for participant i
- v_j = value j

Situations where this is not the case are noted at the time of presentation. Other tools, such as box-and-whisker charts are also employed to illustrate ranges and diversity in the data. The reported standard error values are calculated from the

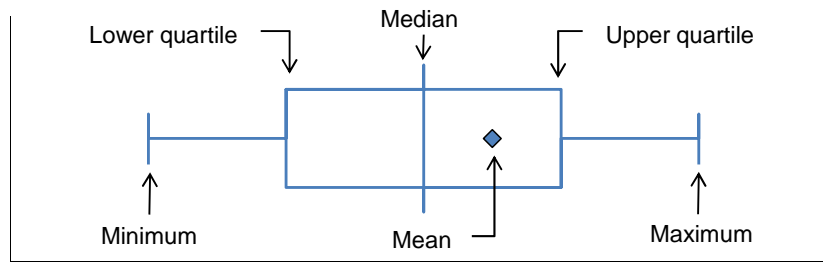


Figure 5.1: Example box-and-whisker chart

user means.

Some analyses also maintain the distinction between the source application of the data: either Microsoft Word or Adobe Reader. This allows a comparison between the tool sets available in each of these applications and the purposes to which they are applied—Word users can create, edit and read documents, while Reader users can only read and inspect documents. The distinction is *not* intended to try and rate one as “better” than the other—such a comparison would be superficial, given the differences in the applications.

Box-and-whisker Charts

Box-and-whisker charts provide a succinct visual summary of the points in a dataset, including minimum, maximum and quartile values. These are especially useful when users show diverse behaviour. For clarity, Figure 5.1 shows the values that are used on box-and-whisker charts in this thesis. The minimum, lower quartile, median, upper quartile and maximum values are displayed, some charts will also illustrate mean values with the use of a diamond.

Highly Repetitive Actions

Researchers have shown that many aspects of human behaviour are heavily biased toward a small percent of possible actions. Many of the analyses presented in this chapter show a similar trend, with a rapid, non-linear decrease in the y-axis, as the x-axis increases. This type of selection has previously been modeled by principles such as Zipf’s law [237], the Pareto principle [117] and the 80–20 rule. For brevity, some analyses presented in this chapter simply report the correlation co-

efficient (R^2) for the fitted power-law or logarithmic curve—relations to previous laws describing such trends are not repeated.

Frequency Analyses

Finally, to ease summarisation and to smooth outliers, frequency reports are often grouped into logical units. For example, in Figure 5.8, the distances travelled are grouped into ranges one page in size. Few navigation actions move an exact number of pages at a time, so each group includes many intermediate values. These ranges are half-open, meaning they exclude the first value and include the second value. Mathematically, the page range 0–1 would be denoted $[0, 1)$.

5.5 Document Properties

Document navigation patterns are influenced by the document under inspection. For example, the scrollbar serves little purpose in a five line document. To set the foundation for the analyses of document navigation, this section presents various properties of the documents used.

5.5.1 Scale of the Study

The number of documents opened during the study provides context for later analyses. Users opened a mean of 175 documents (s.d. 124.5) in Word and 122 documents (s.d. 46.5) in Reader over the 120 day study period (Table 5.2, line 3). The mean number of documents opened per day was four for both Word and Reader (line 7). This is approximately one every two hours of a working day, per application, or one an hour between both interfaces. One Microsoft Word participant only recorded one document with interaction, so is removed from any further analysis.

In these documents, Word users, in total, navigated through approximately 26,700 pages (a mean of 2053 pages per person) and Reader users approximately 29,600 pages (a mean of 2117 pages).

5.5.2 Do users interact with the documents they open?

A surprisingly high proportion of documents were opened and closed without any intervening navigation: 37% in Word and 16% in Reader (line 5). In Word, this

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
<i>General</i>										
1 # of participants					13					14
2 Total # docs. opened					2271					1706
3 # docs. opened, per user	174.7	163	124.5	37	384	122	46.5	183	5	718
4 % docs. that are unique [†]	44.3	44.4	18.3	14.2	75.6	60.4	57.7	26.9	12.0	100
5 % of docs. with interaction	63.4	63.8	9.8	51.1	89.2	83.9	82.3	9.4	69.9	100
<i>Documents used</i>										
6 # days applications used	43.1	45	21.7	16	81	27.0	24	16.0	4	55
7 Mean # docs. opened per day	3.8	3.3	1.7	1.8	7.3	3.9	2.5	5.1	1.3	21.1
<i>Document open and interaction time</i>										
8 Mean doc. open time (mins.)	456.5	320.8	400.0	1.2	1195.6	366.1	114.1	517.8	4.1	1433.9
9 Mean doc. interaction time, DUS idle = 5 mins.	19.2	21.2	9.1	1.1	32.5	6.4	5.9	3.3	2.5	14.0
10 Mean % time interacting, DUS idle = 5 mins.	70.0	68.2	14.3	48.3	98.9	78.1	79.2	14.6	49.7	100

[†] Some document titles were unavailable, so these are omitted from this analysis

Table 5.2: Summary of document use and re-use (*Med.* = median, *s.d.* = standard deviation). Daily statistics based on the number of days where AppMonitor recorded events for that application.

is best explained by the default blank document that is opened as the application starts. If the user then opens the required document using the application’s built-in mechanisms, the blank document is discarded (and will have no user interaction). In Reader, this is most likely caused by users opening an incorrect document.

5.5.3 How often are documents re-opened?

Documents are commonly re-used—approximately half were reopenings of those previously viewed by the user (line 4). This re-use indicates greater user interest in certain documents, that in turn can lead to increased familiarity (potentially influencing the choice of navigation tools).

The document’s title is recorded as it is opened and is used to determine document uniqueness¹. All documents are treated with equal saliency—unsaved “scratch-pad” type documents are not differentiated from full documents with ex-

¹This measure is not 100% accurate—a document in a different directory with the same name is recorded as the same document. However, it is believed this is a relatively rare event and would not significantly influence the results.

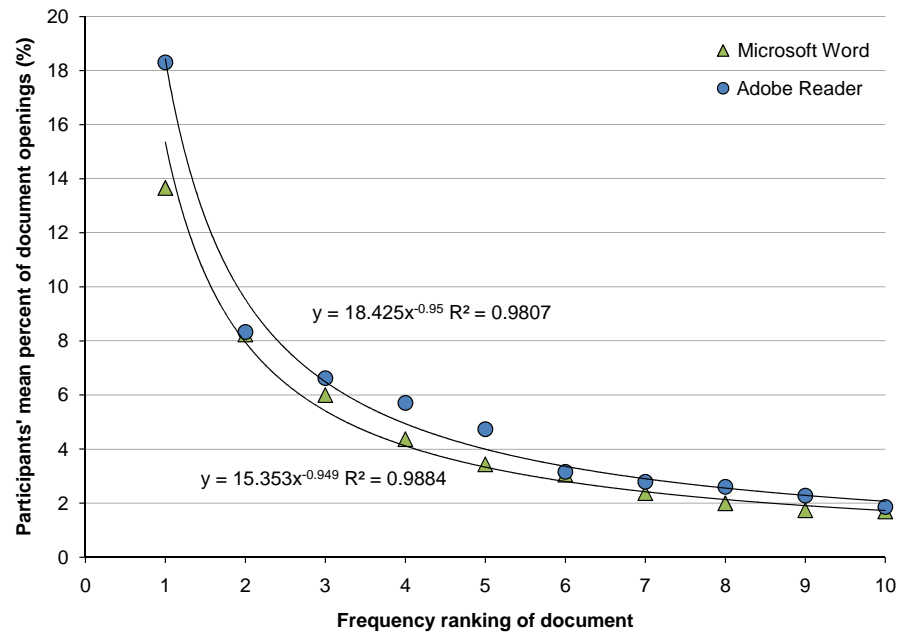


Figure 5.2: Mean distribution of document re-openings for the participants' ten most frequently opened documents (one is the most frequently opened, ten is the tenth most frequently opened). Error bars omitted for clarity.

tensive interaction.

For some users, there was a high probability that a document was a re-opening: both interfaces at had least one user with less than 15% of their openings attributed to unique documents. In contrast, one user had a 100% uniqueness measure. However, this was due to this user only ever opening five documents in Reader.

The mean usage distributions for the ten most frequently opened documents are shown in Figure 5.2. Both applications follow a power-law relationship between the percentage of times a document was opened and the document's frequency rating. On average, the most frequently used document accounts for 14% of Word openings and 18% of Reader openings.

5.5.4 How many pages are there in a document?

The mean length of a Microsoft Word document was 6.8 pages (s.d. 4.3 pages) and an Adobe Reader document was 38.1 pages (s.d. 35.6 pages). Figure 5.3 illustrates the large difference in distributions of the participants' mean document lengths,

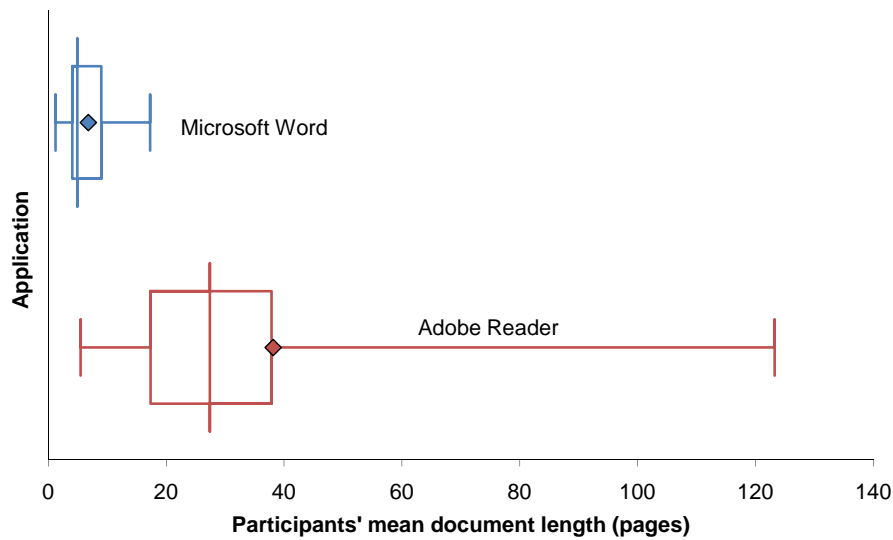


Figure 5.3: Distribution of the participants' mean document lengths, by application

for the two applications.

The longest observed document was 160 pages in Word; 1743 pages in Reader. Such extremities in document length mean that designers of systems that require navigation should be acutely aware of the size of the expected content. For example, when opening a large document, the thumb on a standard scrollbar quickly reaches its minimum size and thereafter a one pixel movement in the thumb can result in a movement of several pages in the document, rendering it useless for fine scale adjustments. Designers should consider whether the default scrollbar is always the correct tool for the application, especially when large variances in the size of the document are expected.

5.5.5 *How long are documents open?*

Documents are generally open for long periods of time. Word documents were open for a mean of approximately 8 hrs and Reader documents for 6 hrs (line 8). However, the large standard deviations and very large minimum–maximum ranges indicate significant variance between participants. A large portion of this time is accounted for documents that are left open on the users' computers overnight (which is explored in the next section).

5.5.6 *How long do users spend interacting with their documents?*

Document interaction time is a direct measure of user interest—large periods of interaction show high interest, short periods show low interest. Recall from Section 5.4.1 that Document Usage Sessions (DUS's) are a measure of the time a document is in use. The sum of the length of all DUS's in a document provides the document interaction time.

As expected, users spend longer interacting with documents that they are authoring (in Microsoft Word), than those which they are searching or reading (in Adobe Reader). The users' mean interaction times, by application, are shown in Figure 5.4.

A comparison between the mean interaction times (Figure 5.4) and the mean document lengths (Figure 5.3) shows that Word documents are generally shorter, but have a longer period of interaction, while Reader documents are generally longer, but have shorter interaction times. This correlates with the tasks users are most likely performing in these applications—document creation in Word, and reading or searching in Reader. One conjecture from these results is that Reader's longer documents are mainly used for reference or partially read and are not open for long periods of time for complete “cover-to-cover” reading.

5.6 **Vertical Document Navigation**

Vertical document navigation—for example, scrolling from the start of the document to the end—is the most commonly performed navigation action. This is not surprising, given that documents are usually presented in the viewport in a linear manner, having a well-defined beginning- and end-point. Users then move within these extremities to view the document's content. This section describes the vertical navigation observed. First, consideration is given to how a characterisation of navigation should be constructed. Second, vertical navigation actions are summarised in three categories: continuous movement, paging and jumping. Third, using these categories, the following attributes are reported: the distribution of user actions across tool-sets, the distances moved, the time spent using the tools and the velocities of the actions.

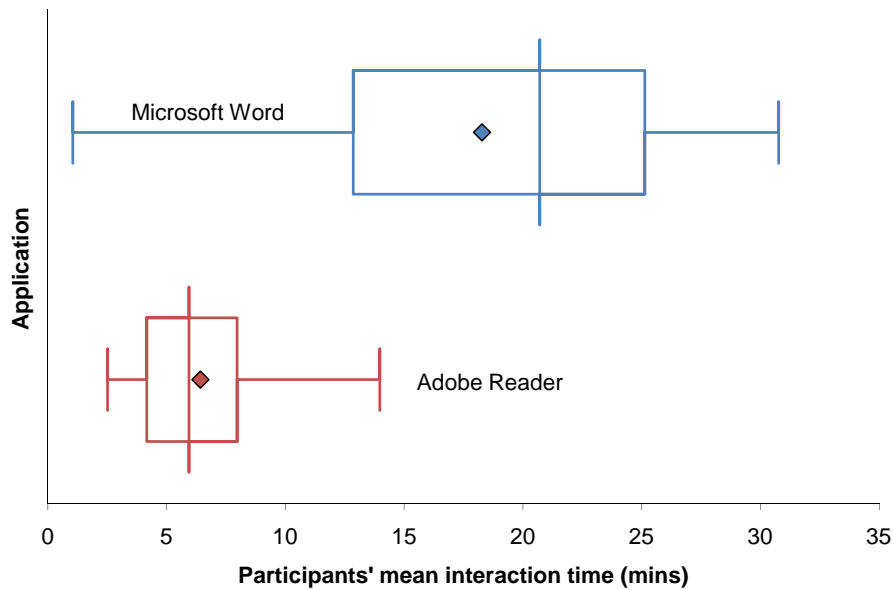


Figure 5.4: Distribution of participants' mean document interaction time, by application

5.6.1 How should vertical document navigation be characterised?

Nearly 37,000 vertical document navigation actions were recorded during the 120 day study period. Each navigation action (for example, dragging the scrollbar thumb) has a series of attributes that describe the action. These attributes are:

- The navigation tool used (for example, the *page down* key).
- The distance moved—measured both as an absolute distance, in pages, and as a percent of the document's length.
- The direction of movement (forward or backward through the document). Some tools allow the direction of navigation to change during an action (for example, when dragging the scrollbar thumb). During analysis, these actions are broken into their uni-directional components (see Section 5.4.3).
- The time duration of the action.
- The velocity of movement (derivable from the distance and time).

- The type of movement: continuous movement, jumping to a position or a paging action (described in the following section).

For completeness, the characterisation presented in this section encompasses all of these properties.

5.6.2 *A Categorisation of Navigation Actions*

The applications under study provide numerous tools to support vertical movement. To aid summarisation, it is helpful to divide these tools into three categories, based on the type of movement they best support. These three categories are:

Continuous movement: Navigation follows a smooth, uninterrupted linear path through the document. If the user wishes to move from point *a* to point *b* (a large distance later in the document) all of the content between these two points in the document is viewed (even if only momentarily). Note that some tools, such as the scrollbar thumb, allow very high velocity movement between two points. In this case, the majority of the intervening content is not displayed to the user (as the system cannot keep up with the rate of information change). For this analysis, tools are classified according to the type of movement they *best* support (so the scrollbar thumb is classified as a continuous movement tool).

Paging: Navigation through the document is linear; however, the content is presented to the user in ‘chunks’ (possibly complete pages). When a navigation action occurs, the information on-screen (the current chunk) is completely replaced with the previous or next chunk. In a similar manner to continuous movement, all document content between points *a* and *b* is viewed.

Jump: The user moves directly (‘jumps’) from point *a* to point *b* without viewing the intervening document content.

These three categories are sufficient to encompass all of the tools available in Microsoft Word and Adobe Reader. However, further categories would be required to cover all types of navigation. For example, a tool that animates the transition between points during a jump action overlaps two of the categories presented here.

Continuous	Paging	Jump
Arrow keys	Paging UI buttons	Bookmarks
Hand-tool (Reader only)	Paging keys	Find
Highlight-drag	Scrollbar trough	First & last page shortcuts
Mousewheel [†]		Goto
Rate-based scrolling		Internal link
Scrollbar arrows [†]		Thumbnails
Scrollbar thumb [†]		View navigation buttons

[†] In Adobe Reader's *single page* mode the mousewheel and scrollbar arrows may act as paging tools if such a zoom is selected that a complete page is visible on-screen. The scrollbar thumb may act as a jump tool if the same condition is met.

Table 5.3: Categorisation of Microsoft Word's and Adobe Reader's document navigation tools

5.6.3 A Categorisation of Word and Reader's Tools

The tools available for *explicit*² vertical document navigation in Microsoft Word and Adobe Reader are shown, by category, in Table 5.3. The continuous and jumping categories have more than double the number of tools available for paging.

5.6.4 How are users' actions divided between the navigation categories?

The majority of navigation is performed using continuous navigation tools, making up 84% of actions across both applications (see Figure 5.5). Jump actions make up 14% and paging just 2%. However, the number of actions is not the only measure of navigation—the following sections will cover the distance, period and velocity of actions.

Adobe Reader users show greater variability between the navigation categories, when analysed on a per-user basis. All Microsoft Word users recorded over 70% of their actions using continuous navigation tools. Conversely, Adobe Reader users were more variable, with several users have large portions of their actions made up of jump actions, as illustrated in Figure 5.6.

²See Section 5.4.2.

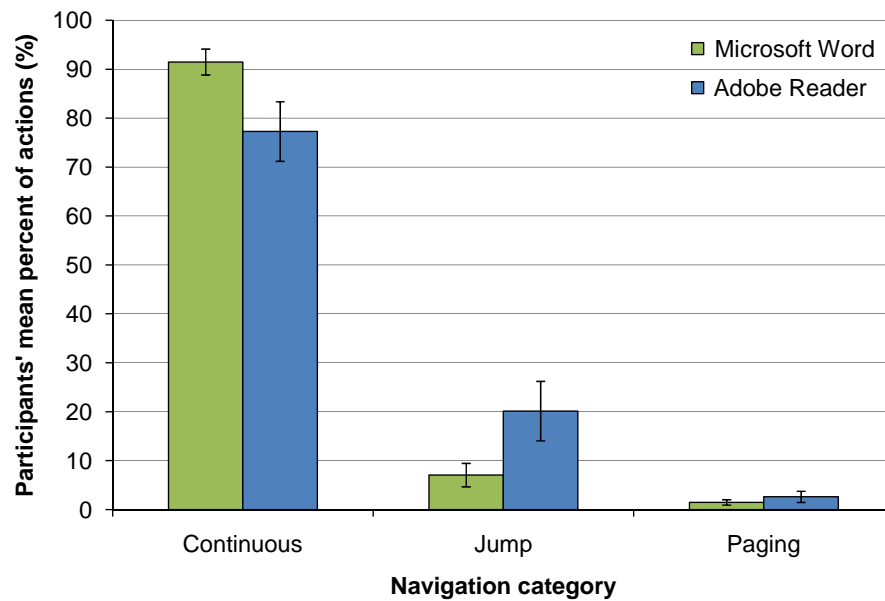


Figure 5.5: Distribution of navigation actions into categories. Error bars show standard error.

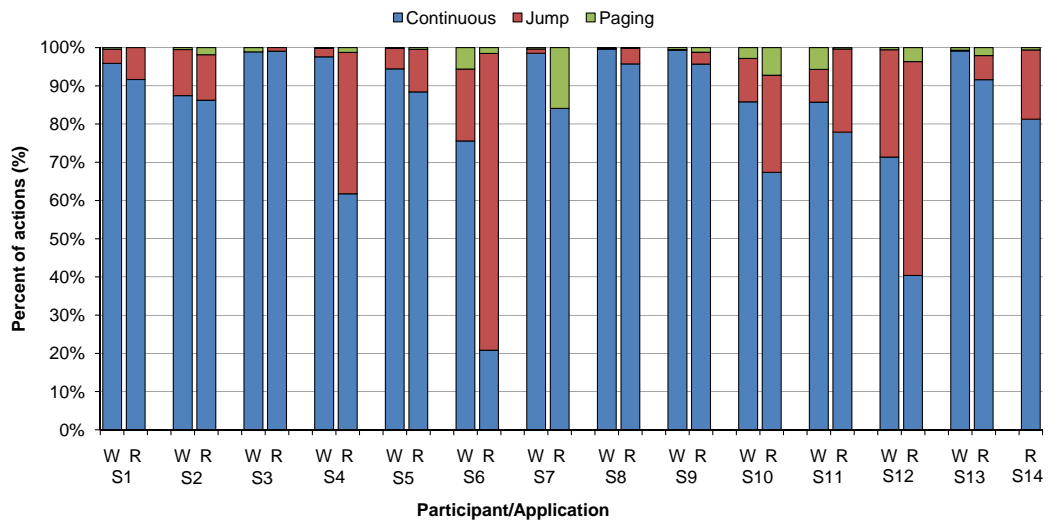


Figure 5.6: Per-user categorisation of navigation actions, per application (*W* = Microsoft Word, *R* = Adobe Reader)

5.6.5 Are users' actions consistent across applications?

Seven of the thirteen participants who used both applications show consistent (within 10%) use of the navigation categories across both Word and Reader (see Figure 5.6). The remaining users show substantial differences in the percent of actions falling into each category. For users who show consistency, their actions can safely be generalised to other applications. Interaction designers will find it more difficult to accurately model and generalise the navigation behaviours of the participants who employ different strategies across applications.

5.6.6 How do individual tools contribute to each navigation category?

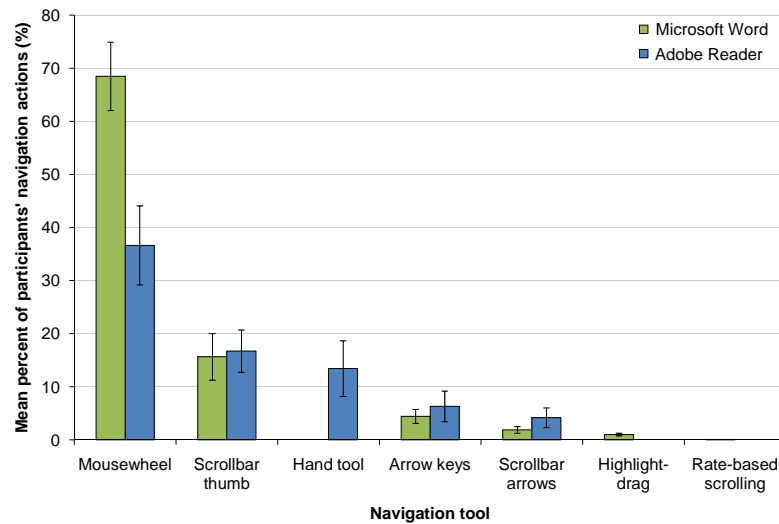
The mousewheel, scrollbar thumb and paging keys are the tools employed for the majority of navigation. The contribution of each tool to the three categories is shown in Figure 5.7 (note that each graph has significantly different scales). The mousewheel contributes the greatest number of actions, especially in Microsoft Word, where it makes up 68% of all navigation. The scrollbar thumb and paging keys are the next most dominant tools.

As expected, tools that jump to different document positions individually contribute to less than 1.5% of all actions. Users of these tools have a good idea of their target position. Continuous and paging tools are potentially employed when the user wishes to visualise each page of the document (possibly for visual search tasks). More continuous or paging actions may be required to achieve the same target that is reached with a single jump action.

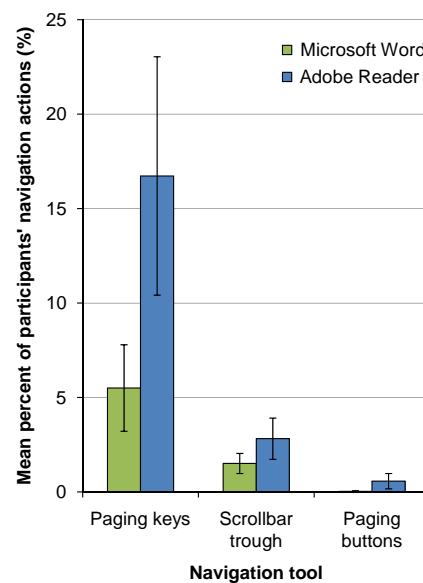
5.6.7 How far do navigation actions move?

The majority of navigation actions move small distances, with approximately 75% moving less than one page at a time. A breakdown of the mean vertical distance moved by individual navigation actions is shown in Figure 5.8. Negative distances represent movement backward (toward the beginning) and positive distances represent movement forward in the document (toward the end).

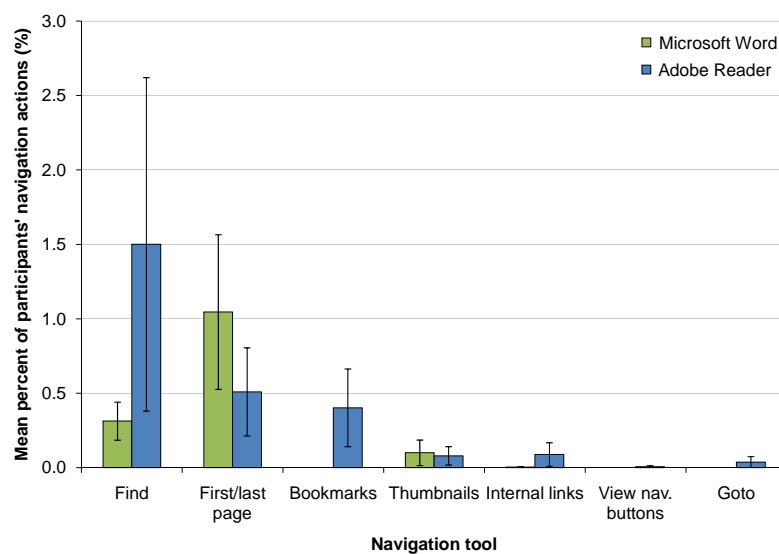
It remains to be seen whether this result would change if more efficient long distance navigation tools were deployed. Many researchers have focused their attention on designing systems that are efficient at achieving targets a large distance from the starting position. Further research is required to determine if small



(a) Distribution of continuous tool actions. Note, rate-based scrolling accounted for 0.06% of Word's actions and none of Reader's actions.



(b) Distribution of paging actions. Note, Word's paging buttons accounted for 0.03% of actions.



(c) Distribution of jumping tool actions. Note, the view navigation buttons accounted for 0.007% of Reader's actions.

Figure 5.7: Distribution of navigation actions, by tool. Error bars show standard error. Note the significantly different scales across navigation categories.

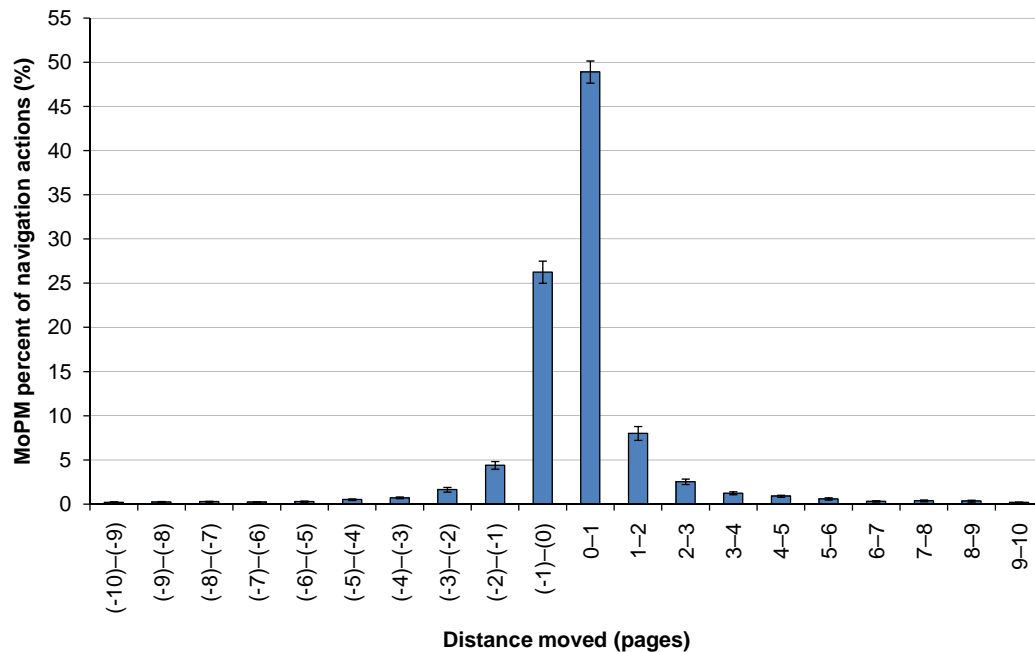


Figure 5.8: Distribution of the mean distances moved by navigation actions. MoPM = Mean of Participants' Mean (see Section 5.4.4). The graph displays a mean of 98% of actions, with error bars showing standard error.

movements are combined together to achieve a long distance target or if users predominately require small distance navigation.

Forward movement dominates backward movement, accounting for 64% of actions. This result is expected, as many tasks, such as proof-reading, require a predominantly linear path toward the end of the document, with no necessity to return to the beginning before closing.

Paging actions move further, per action, than their continuous counterparts. Figure 5.9 shows the mean distance navigated, by category. Comparing Figure 5.9 with Figure 5.5 shows paging actions make up only a small percent of actions, but a significantly larger percent of the mean total distance navigated.

To understand how each of the navigation tools contribute to these categories, a further breakdown of the percentage of total distance travelled, by tool, is shown in Figure 5.10. The mousewheel, scrollbar thumb and paging keys dominate over all other tools. The remaining techniques moved significantly less distance than these three. The actions that make up these distances are broken down further in

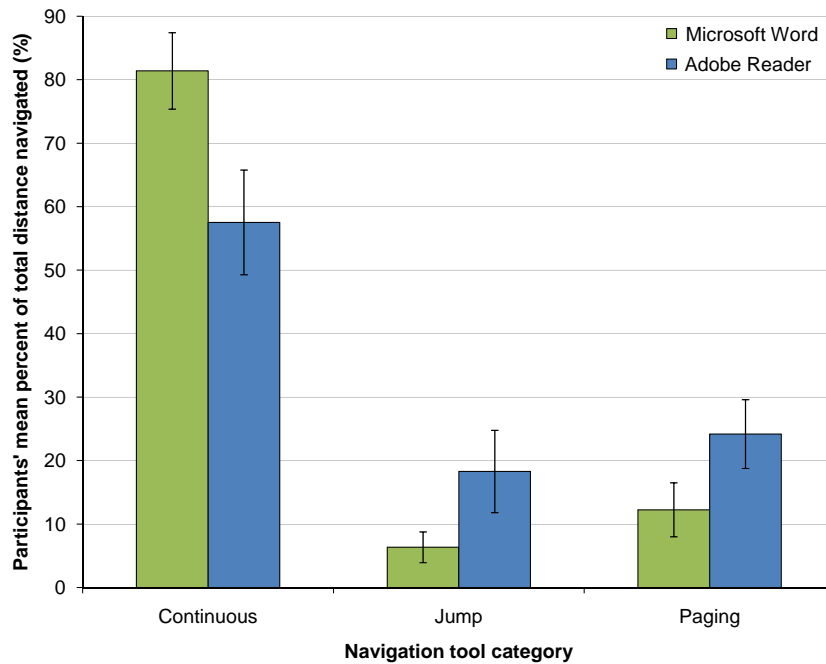


Figure 5.9: Mean percent of total distance navigated, by category. Error bars show standard error.

the individual tool analyses of Section 5.7. Another important factor to consider when navigating is the time duration of each action.

5.6.8 What is the time duration of navigation actions?

The total time duration of an individual action is comprised of the cognitive time to initiate the action, the physical movement time to bring appropriate limbs to the input device, the time to correctly position the device and the time required to manipulate controls or the document itself. AppMonitor can only record the last of these—the interface interaction time—so all values would, in total, be larger than those reported. Figure 5.11 summaries the time duration of all vertical navigation actions.

Navigation actions are generally short, with over half of all actions taking less than one second to complete and over 75% less than two seconds. The drop-off in percent of actions as the time duration increases is closely modelled by an exponential decay, with correlation of $R^2 = 0.98$. This rapid drop-off can be attributed

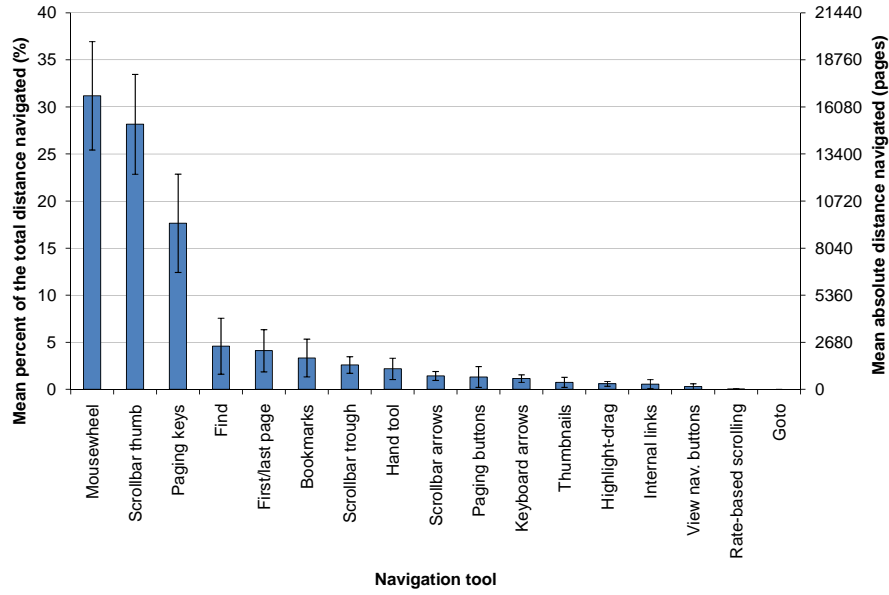


Figure 5.10: Mean distance navigated, by tool. Error bars show standard error. Note, rate-based scrolling accounts for a mean of 0.04% of the distance navigated and goto actions for 0.003%.

to reading, writing and proofing activities where short distance movement is sufficient (as in Figure 5.8) resulting in short periods of navigation. Further, when users do make long duration actions, that may not be uni-directional—recall that this analysis splits those actions into their constituent components.

5.6.9 What percent of users' time is spent navigating?

The knowledge of the time period of individual actions is used to determine the total time spent navigating in a document. This is calculated by summing the periods of navigation actions and dividing by the user's interaction time with the document:

$$t_{nav} = \frac{\sum_{i=0}^n t_i}{t_{int}} \times 100 \quad (5.2)$$

where:

$$t_{nav} = \text{percent of time navigating in document}$$

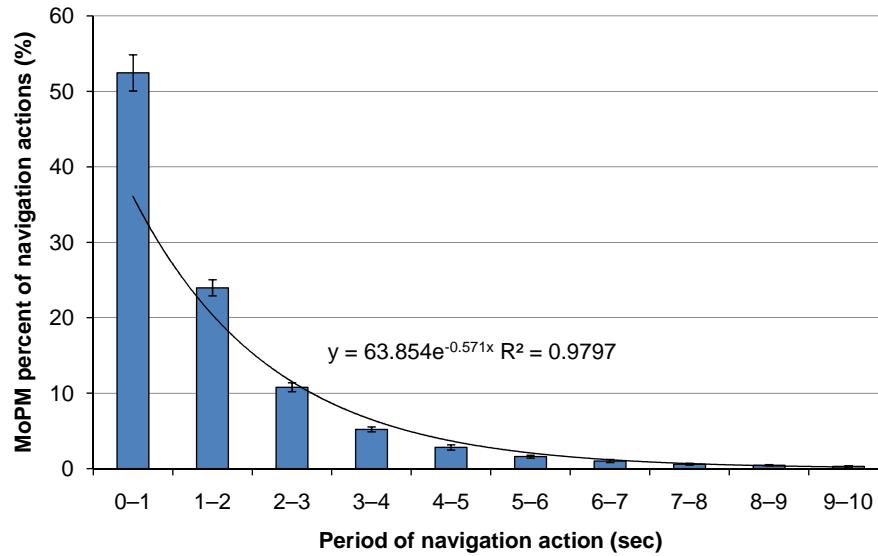


Figure 5.11: Mean time period of navigation actions. The graph is truncated at 10 seconds and shows a mean of 99% of actions. Error bars show standard error.

n = number of navigation actions in document

t_i = period of navigation action i

t_{int} = total interaction time with the document

Adobe Reader users spend a greater percent of their time navigating (mean 13%) than Microsoft Word users (mean 6%), as shown in Figure 5.12. The majority of users spent on average, below 20% of their time navigating. This result aligns closely to Byrne et al.’s observation of the length of time spent navigating in web-browsers. Their study found that “approximately 40 minutes in our 5-hour sample was spent scrolling” [37, pg. 550], equating to around 13% of the users’ time.

5.6.10 What is the velocity of navigation actions?

Navigation velocity, the speed with which a user moves through the document, varies significantly between the navigation categories. Continuous scrolling methods (Figure 5.13a) have the slowest mean velocities, averaging 0.22 pages/sec. Paging tools (Figure 5.13b) average 0.64 pages/sec and jump tools (Figure 5.13c)

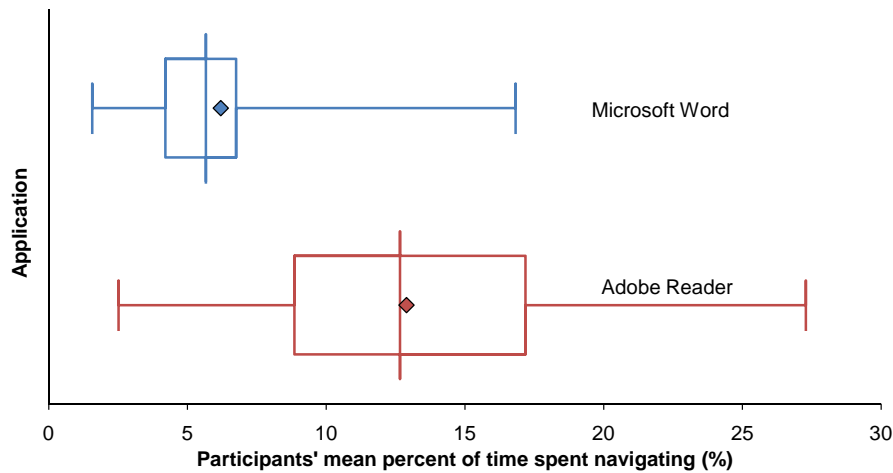


Figure 5.12: Mean percent of participants' interaction time spent navigating

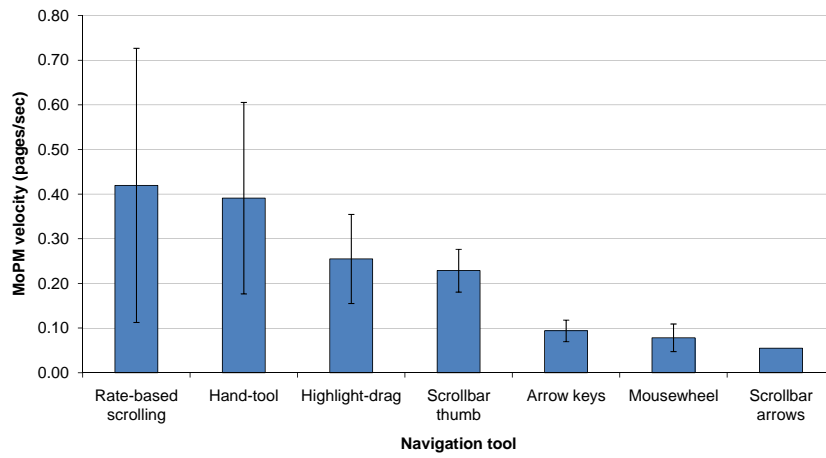
a significantly higher 39.38 pages/sec. These differences are explained by the types of actions taking place. Paging actions generally move at least a page (see Section 5.7.3 for more details), whereas many continuous tools better support short distance movement. Jump tools can move vast distances (hundreds of pages) in the same time it takes to move a single page, often with the click of a button. For this reason, velocity is a poor between-category comparison measure.

Further, many of the tool velocities reported in Figure 5.13 have very large standard error bars. This is indicative of the large differences in the users' mean velocities when manipulating these tools. This demonstrates tool flexibility, allowing the user to move at a speed comfortable to their task.

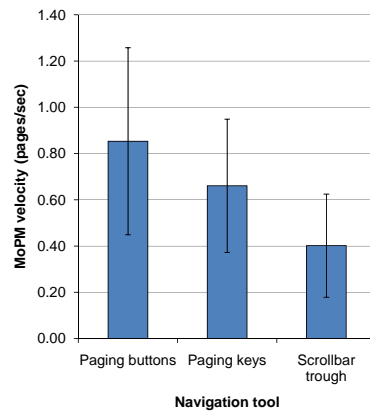
5.7 Analysis of Selected Vertical Document Navigation Tools

The previous section provided an overview of the document navigation tool use during the longitudinal study. This section examines in more depth the use of the mousewheel, the scrollbar thumb, the paging tools, Reader's panning tool, bookmarks, find and goto tools and rate-based scrolling. It also classifies users into stereotypical navigator categories.

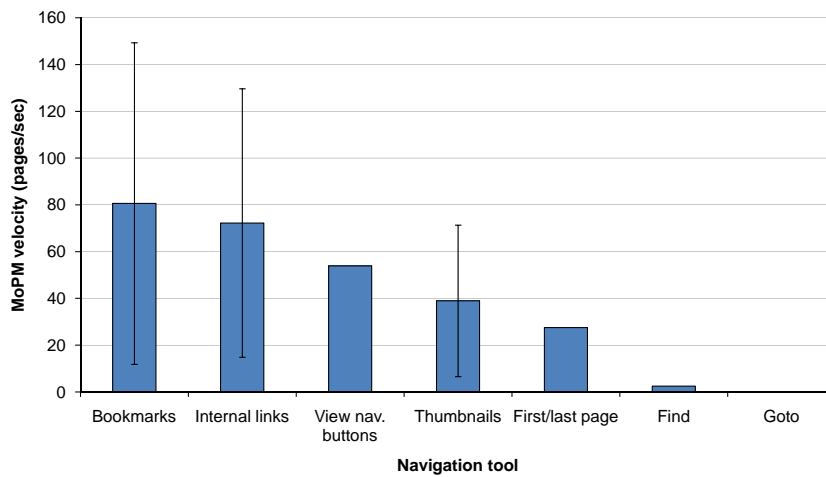
To begin, a summary of the three most used tools is shown in Figure 5.14. In this figure, the mean time duration of navigation actions is shown on the vertical



(a) Mean velocities for continuous navigation actions



(b) Mean velocities for paging actions



(c) Mean velocities for jump actions

Figure 5.13: Mean navigation velocities, by tool. Error bars show standard error (tools without error bars were only employed by a single user). Note the significantly different scales across navigation categories.

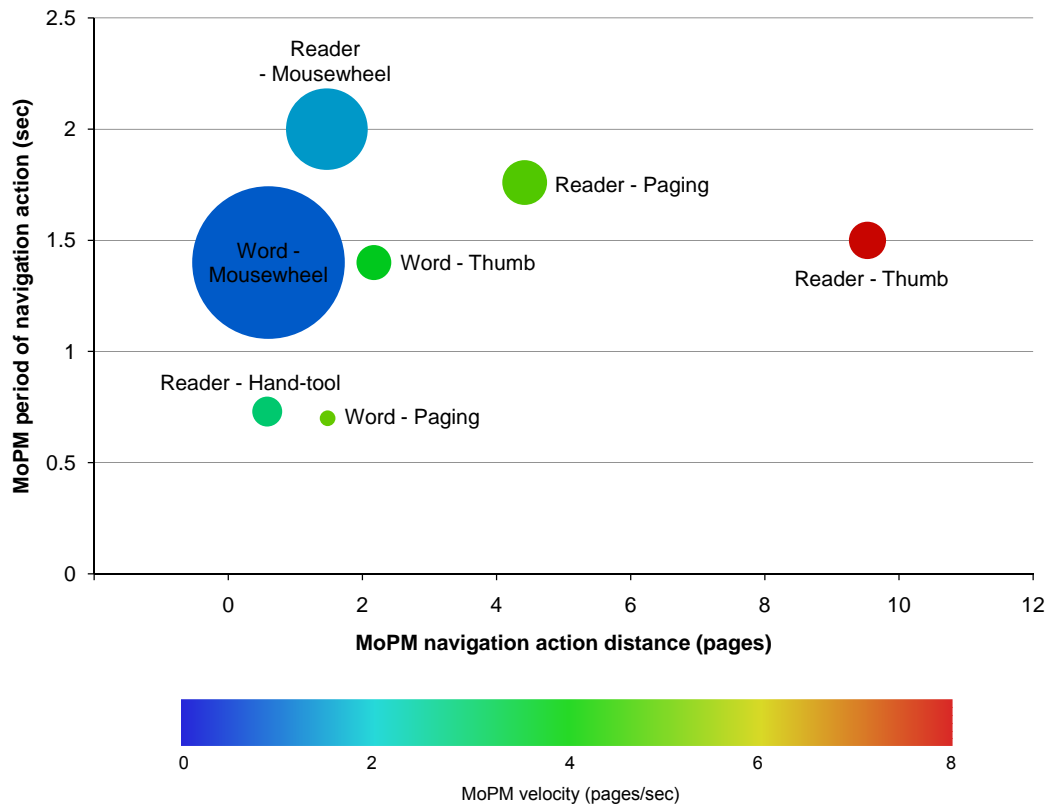


Figure 5.14: Summary of the attributes of navigation tool use. Bubble width illustrates the relative number of actions. Bubble colour indicates velocity, with the scale below the graph.

scale, the mean distance on the horizontal scale, the colour indicates mean velocity and the relative number of actions is approximated by the bubble width.

5.7.1 The Mousewheel

The mousewheel is the most frequently used navigation tool, accounting for 68% of Word and 36% of Reader's navigation actions (Figure 5.7a). The mousewheel also accounted for the greatest mean total distance moved (Figure 5.10). Further summary statistics for the use of the mousewheel are reported in Table 5.4.

The mousewheel was used by all Microsoft Word users and all but one Adobe Reader user (Table 5.4, line 1). It was used in 54% of Word documents and 45% of Reader documents (line 2). Adobe Reader mousewheel actions had a mean

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
1 # participants who used					13/13					13/14
2 % of docs. used in	53.8	65.6	26.6	3.8	81.8	45.0	52.5	28.2	7.3	95.7
3 # actions/doc.	12.7	13	10	0.2	35.8	2.9	2.5	2.4	0.2	7.3
4 Distance per action (pages)	0.6	0.5	0.3	0.2	1.1	1.5	1.6	0.8	0.5	2.8
5 Time period of actions (sec)	1.4	1.4	0.3	1.0	2.1	2.0	2.0	0.5	0.8	2.7
6 Velocity of actions (pages/sec)	0.9	1.0	0.3	0.4	1.6	1.5	1.2	1.8	0.3	7.1

Table 5.4: Summary statistics for the use of the mousewheel

distance nearly a page larger than those in Word (line 4). This is accounted for by the use of Reader's single page mode, where a single 'click' of the mousewheel may cause a large distance movement as the next page comes into view. Actions were also longer and faster in Reader.

Mousewheel actions are also short, with a mean of 31% of actions taking less than half a second (see Figure 5.15). These actions represent users moving a few clicks of the mousewheel, to navigate up or down a few lines in a document. Ninety-five percent of mousewheel actions take between 1–5 seconds.

How long do mousewheel actions take to move a specified distance?

The mean time required to scroll a specified distance using the mousewheel is modelled by a non-linear relationship. This relationship is shown in Figure 5.16. This trend holds for the page range 0–2 pages (as shown); however, outside of this range, movement times become more diverse and do not fit this pattern (or show any reliable relationship). The non-linear relationship is attributed to the different tasks-at-hand: when moving small distances (< 0.5 pages), users move slowly as they are reading or accurately positioning the document, using only a few clicks of the wheel. For larger distances (> 0.5 pages) a flicking action is more probable, with users aware that their target is more than a few paragraphs away.

The actions recorded in this study consist of both concise, targeted actions (where the user is completely familiar with the document and knows exactly where they are navigating), imprecise search actions (where the user does not know where their target is, or even what it is) and a those on some scale in-

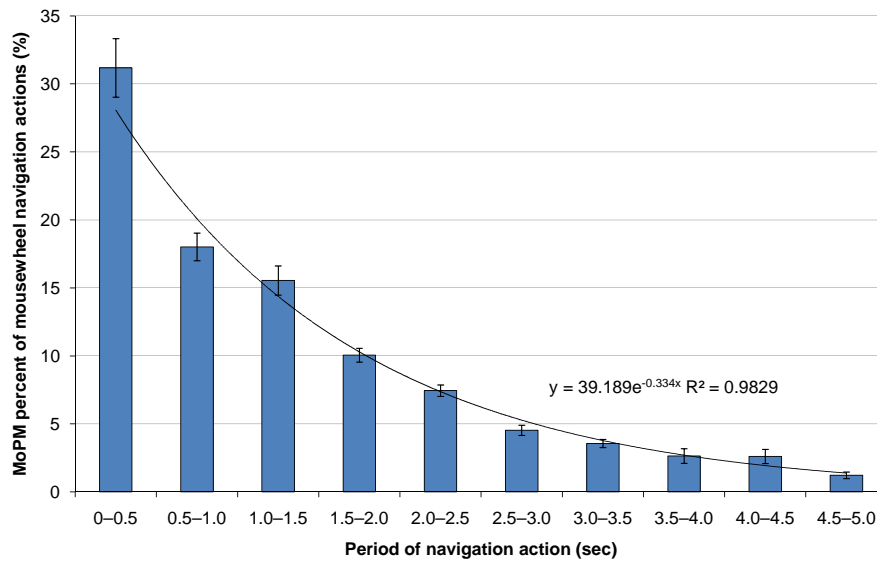


Figure 5.15: Period of mousewheel navigation actions. Error bars show standard error. The graph is truncated at five seconds, which includes an average of 95% of all actions.

between. This wide variety actions makes the data-set ideal for modelling average user performance with this tool.

5.7.2 The Scrollbar Thumb

The scrollbar thumb was used for 16% of Microsoft Word and 27% percent of Adobe Reader's navigation actions (see Figure 5.7a). It is the second most commonly used tool, behind the mousewheel. Summary statistics for the use of the scrollbar thumb are provided in Table 5.5. All participants used the scrollbar thumb in an average of 28% of Word documents and 33% of Reader documents (lines 1-2). Word users had an average of 4.5 scrollbar thumb actions per document and Reader users 2.1 actions (line 3). Scrollbar thumb actions have a mean time duration of 1.4 sec in Word and 1.5 sec in Reader (line 5)

How far do scrollbar thumb actions move?

Distances travelled with the scrollbar thumb are much larger than those observed with the mousewheel. Word users travelled an average of 2.0 pages per action and

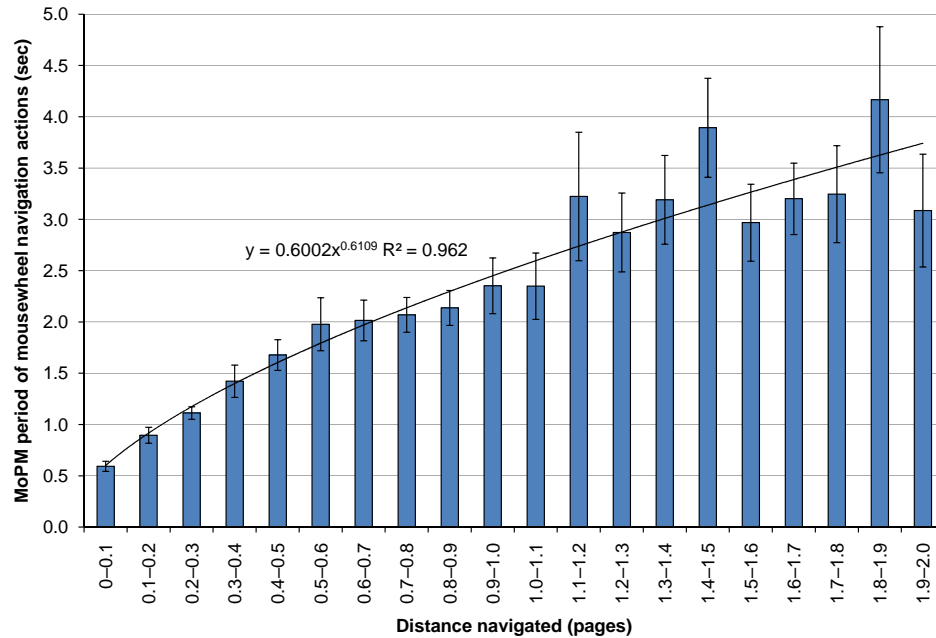


Figure 5.16: Correlation between period and distance of mousewheel navigation actions between 0–2 pages. Error bars show standard error.

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
1 # participants who used tool					13/13					14/14
2 % of docs. used in, pp.	28.1	19.0	20.5	6.3	71.2	32.9	32.7	24.7	3.6	87.5
3 # actions/doc.	4.5	1.1	7.7	0.1	26.7	2.1	0.7	3.3	0.1	12.6
4 Distance per action (pages)	2.0	1.8	1.4	0.19	4.5	9.5	4.9	13.3	1.1	51.1
5 Time period of actions (sec)	1.4	1.4	0.3	1.0	1.9	1.5	1.5	0.7	0.3	2.9
6 Velocity of actions (pages/sec)	3.7	2.3	3.8	0.2	13.2	8.0	4.4	11.0	0.5	43.7

Table 5.5: Summary statistics for use of the scrollbar thumb

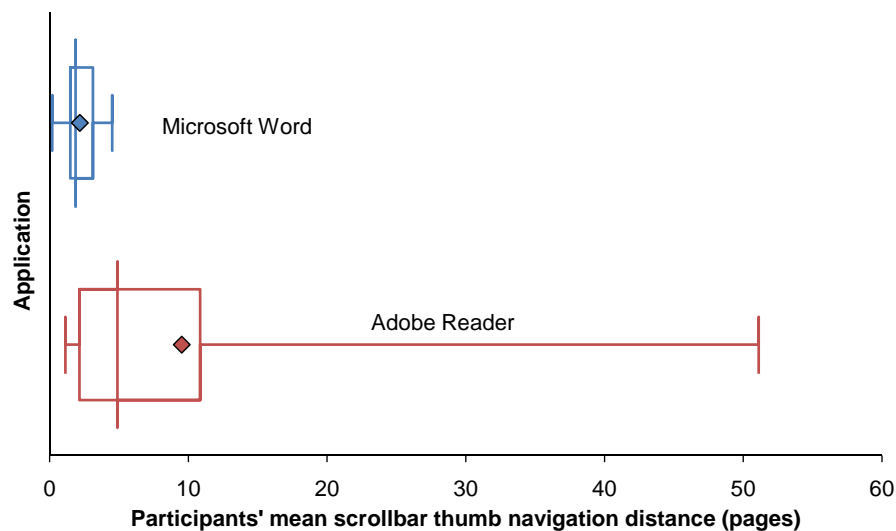


Figure 5.17: Participants' mean distance navigated with the scrollbar thumb

Reader users an average of 9.5 pages per action (Table 5.5, line 4). In comparison, mousewheel actions travelled 0.6 and 1.5 pages per action. The largest single scrollbar thumb drag in Word was 48 pages and in Reader was 680 pages.

Participants showed large variances in their use of the scrollbar thumb. A box-and-whisker plot of participants' mean distance moved with the scrollbar thumb is shown in Figure 5.17. The shape of these plots is similar to the plots of the documents' length (Figure 5.3)—Word's was small and compact, Reader's was quite disperse. As expected, the distances travelled with the scrollbar thumb are strongly related to the length of the document.

How fast is the scrollbar thumb dragged?

The scrollbar thumb allows the user to vary their velocity through the document—it may be dragged very slowly, extremely fast or at any measure in-between. Microsoft Word users tend to move more slowly with the scrollbar thumb, with an average velocity of 3.7 pages/sec, compared to Adobe Reader users with an average of 8.0 pages/sec. Both of these values are significantly faster than the mean velocities of the mousewheel—0.9 and 1.5 pages/sec respectively.

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
1 # participants who used tool					10/13					10/14
2 % docs. used in, pp.	18.3	5.9	25.4	0.7	73.1	28.4	18.7	29.6	0.9	80
3 # actions/doc.	2.0	0.2	3.9	0.01	12.7	2.9	0.9	3.9	0.01	11.8
4 Distance per action (pages)	1.7	1.9	1.0	0.1	3.3	3.5	3.5	2.5	0.2	9.3
5 Time period of actions (sec)	0.8	0.7	0.4	0.1	1.5	1.7	1.6	10.0	1.8	3.8
6 Velocity of actions (pages/sec)	4.8	5.0	2.0	1.6	7.5	5.9	4.1	5.2	0.9	17.3

Table 5.6: Summary statistics for use of the paging keys

Scrollbar Thumb Navigation Summary

The scrollbar thumb was used by all participants to generally move large distances (an average of 2.0 pages in Word and 9.5 pages in Reader). This movement is usually fast (1.4–1.5 secs/action) and at a high velocity (3.7 and 8.0 pages/sec).

5.7.3 The Paging Keys

The paging keys were the third most used navigation mechanism, accounting for a mean of 5.5% of Word and 16.7% of Reader actions. A summary of the properties of paging keys is provided in Table 5.6. The majority of the participants used the paging keys (line 1), although more rarely than the mousewheel and scrollbar thumb (line 2). However, at least one participant regularly used the paging keys in 73% and 80% of their documents.

The paging keys were used to move moderate distances in Microsoft Word (line 4), sitting between the small distances of the mousewheel and the large distances of the scrollbar thumb. Paging keys were used to move larger distances in Adobe Reader (mean 3.5 pages, line 4), but still less than the average 9.5 pages moved by the scrollbar thumb in the same application. On average, 75% of these paging actions traveled between one and two pages.

Paging key actions in Word are shorter in duration than those in Reader (line 5), a likely by-product of the larger distances travelled in Reader. Single key presses that fall into the 0–0.2 sec time period account for 37% of actions.

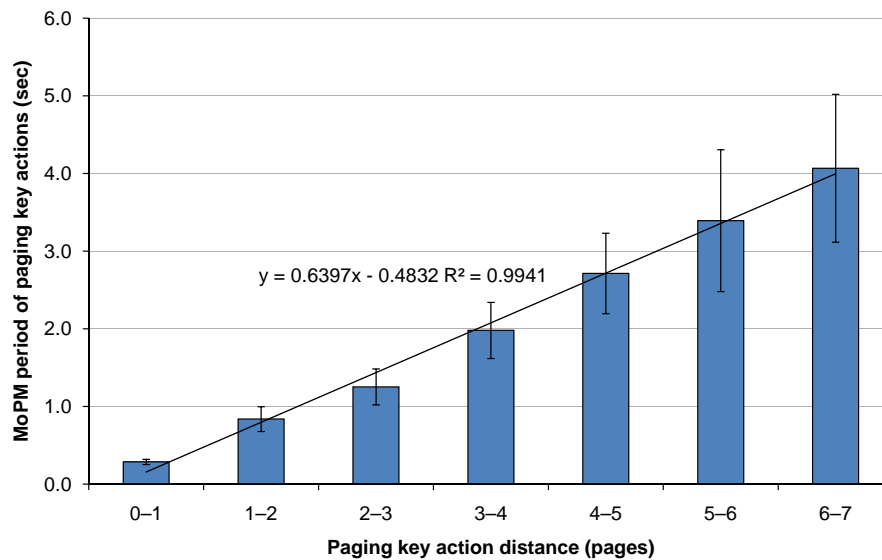


Figure 5.18: Relationship between the period and distance of paging key actions in the range 0–7 pages. Error bars show standard error.

How long do paging key actions take to move a specified distance?

As with the mousewheel, a relationship exists between the period and distance of paging key actions. There is a linear correlation ($R^2 = 0.99$) between these two properties, for values between 0–7 pages, as shown in Figure 5.18. Outside of this range, values become disperse, with no clear relationship. This cutoff is indicative of the time at which users engage the auto-repeat paging functionality.

5.7.4 Panning Using Reader's Hand Tool

The hand tool, available in Adobe Reader, allows the user to pan the view by dragging a page both vertically and horizontally (if available) or in both directions simultaneously. Single pan movements are limited to the number of pixels that a user can drag the mouse over with a single action, in the desired direction. Summary statistics for the use of the hand-tool are provided in Table 5.7. The hand-tool was used by the majority of participants, in 28% of their documents, to move rapidly over short distances.

Hand-tool actions travel an average distance of 0.6 pages, with virtually all navigation (an average of 96.9% of actions) moving less than one page (see Fig-

Analysis	Mean	Med.	s.d.	Min	Max
1 # participants who used tool					12/14
2 % of docs. used in, pp.	28.2	15.6	22.8	4.5	74.4
3 Distance per action (pages)	0.6	0.4	0.8	0.1	3.1
4 Time period of actions (sec)	0.7	0.6	0.3	0.4	1.4
5 Velocity of actions (pages/sec)	2.9	0.9	4.9	0.2	15.9

Table 5.7: Summary statistics for use of the hand-tool in Adobe Reader

ure 5.19). The direct document manipulation supported by this tool (by dragging the face of the document) means that it was primarily designed for moving short distances. Interestingly, the average distance moved is close to the maximum supported on a typical computer setup: a 1680×1050 px resolution screen with Adobe Reader maximised and the document at 100% zoom, means approximately $\frac{2}{3}$ rds (or 0.6 pages) of the document is visible (this calculation may differ depending on the screen size, decorations, toolbars enabled and toolbar sizes). This means at 100% zoom, users can only pan vertically 0.6 pages in a single action. Distances greater than 0.8 pages show a sharp drop-off due to this restriction, indicating users often reach the limits of the distance this tool can move.

5.7.5 Bookmarks

Microsoft Word and Adobe Reader both provide mechanisms to support bookmarking. Word allows insertion, deletion and navigation to bookmarked regions. However, no participants used any of Word's bookmarking features during the study period.

Adobe Reader provides users with a panel for accessing author defined bookmarks. This panel can be open or hidden. The bookmarks panel was visible in 640 of the documents opened by 9 participants. However, one participant accounted for 529 of these (91% of their document openings). On average 27.5% of documents had the bookmarks panel visible at opening. The tab was opened by two participants in four documents (once in each).

Bookmarks were selected 65 times in 41 different documents by four participants; thirty-seven of these documents had the bookmarks panel open when the

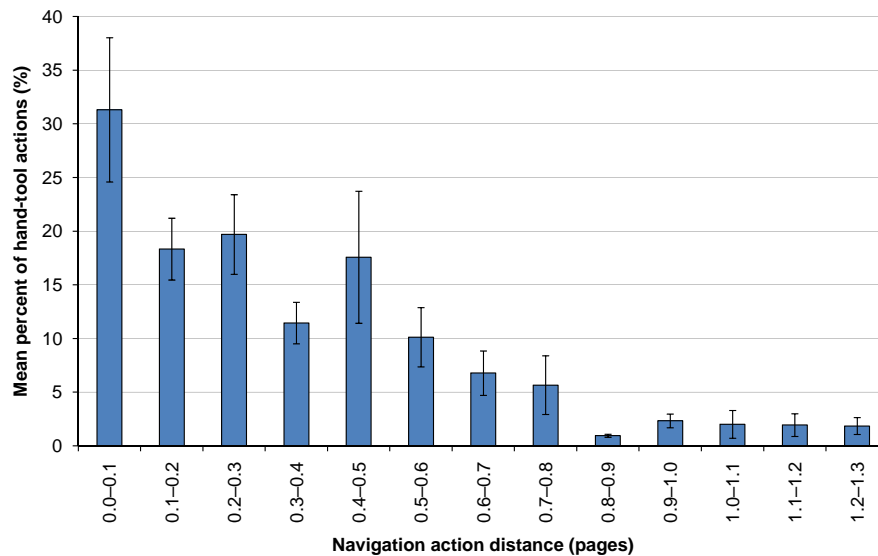


Figure 5.19: Distances of hand-tool navigation actions. The graph is truncated at 1.3 pages and displays an average of 97.9% of all hand-tool actions. Error bars show standard error.

document was opened.

Overall, bookmarks are seldom used. However, use of this tool increases when the panel is open, due to its greater visual prominence.

5.7.6 Thumbnails

In a similar manner to bookmarks, both applications provide thumbnail-enhanced scrollbars to support navigation. Again, this tool is rarely used by participants. A total of 16 thumbnail selections were made in Adobe Reader in 8 documents, by two people. Slightly more use was observed in Word, with 37 thumbnail selections made by four participants, in 21 documents. Similarly to the bookmarks tool, the default ‘hiding’ of this navigation mechanism means it has poor utilisation.

5.7.7 Use of Find and Goto Tools

Find tools allow users to enter a search phrase and have the computer locate occurrences of that phrase in their open document. This is considered one of the key advantages of electronic media over paper [32]. In this study, find tools were

Analysis	Microsoft Word	Adobe Reader
1 # participants who used find	11	10
2 % docs. used in [‡]	5.3 [3.8] ± 4.3 (1.4–14.3)	4.7 [3.1] ± 3.4 (1.3–11.8)
3 # find actions	93	44
4 Method of initiation	<i>Ctrl-f</i> : 92 <i>Edit</i> → <i>Find...</i> : 1	<i>Ctrl-f</i> : 34 <i>Search push-button</i> : 10
5 # searches per action ^{†‡}	5.5 [2.0] ± 14.0 (0–128)	3.4 [2.0] ± 4.6 (0–19)
6 Time from doc. opened to initiation (sec) [‡]	3367.4 [1013.8] ± 5199.6 (1–26264.3)	2218.6 [28.4] ± 12234.6 (1.1–81342.6)

[†] The number of times, for example, the *find next* button is pushed

[‡] Mean [median] ± s.d. (min–max)

Table 5.8: Summary of find actions

occasionally used—93 times in Word and 44 times in Reader (Table 5.8). The relatively little use of these functions backs up Loizides and Buchanan’s [137] findings that find tools are poorly-utilised, despite their apparent popularity.

Only one occurrence of a *goto* action that moved the document position was recorded in an Adobe Reader document. Several other presses of the *Ctrl-g* shortcut were recorded, however, these did not result in navigation actions that moved away from the current document position.

5.7.8 Rate-based Scrolling

Rate-control techniques (that control velocity of movement rather than position) anecdotally polarise users’ opinions—some love them, others hate them. Rate-based scrolling is only available in Microsoft Word, but saw very little use. It accounted for 0.1% of Word navigation actions and was used 27 times, by four participants. One participant accounted for 14 of these uses and one a further seven—the remaining two may have triggered the action “by accident”.

The mean velocity of rate-based scrolls was 0.42 pages/sec (the fastest of the continuous scrolling tools, see Figure 5.13a) and covered 0.03% of the total distance navigated. For this study group the use of rate-based scrolling was negligible; however, due to the polarising nature of this tool, wider studies are needed to determine if this pattern generalises to all users.

5.7.9 Classifying Users into Stereotypical Navigator Categories

The previous sections showed that the mousewheel, scrollbar thumb and paging keys were the dominant navigation mechanisms. However, these tools are not employed identically by all users—some users employ some tools to a greater extent than others. Categorising users’ stereotypical navigation actions would allow interaction designers to present more targeted navigation tools or help to advance users from beginners to experts.

How should users be categorised?

The navigator categories should generalise, but still accurately describe, users’ actions. This classification uses the two tools that recorded the greatest number of actions to define the categories. Two tools were considered sufficient to characterise these users—in Word the two most used tools accounted for 89.2% (s.d. 6.7%) of actions, in Reader 78.1% of actions (s.d. 10.2%). Using two tools allows easy generalisation of this participant set—a very large number of participants would allow a more precise categorisation.

What are the stereotypical categories of user?

Word users in this study are represented by four categories, as shown in Table 5.9. Over half of the participants used the mousewheel and scrollbar thumb for majority of their actions. Reader users were more diverse. Nine users fell into the same categories as Word users, again with the mousewheel/scrollbar thumb combination the most popular. The remaining five users showed no commonality between their tool use. These categories are sufficient to classify the users in this study. Larger studies may require further categories to accurately describe all users’ employment of tools.

Are tools employed consistently across applications?

Earlier, this chapter reported that seven of the thirteen participants consistently applied the same *types* of tools across applications. In this more specific classification of tool use, five participants fell into the same categories across both

Participant	Word				Reader								
	Mousewheel/Thumb	Mousewheel/Paging keys	Mousewheel/Arrow keys	Thumb/Paging keys	Mousewheel/Thumb	Mousewheel/Paging keys	Mousewheel/Arrow keys	Thumb/Paging keys	Mousewheel/Trough	Hand-tool/Mousewheel	Paging keys/Scrollbar arrow	Paging keys/Hand-tool	Thumb/Hand-tool
1	✓				✓								
2		✓							✓				
3	✓									✓			
4			✓								✓		
5	✓				✓								
6		✓										✓	
7	✓												✓
8	✓				✓								
9			✓		✓								
10			✓			✓							
11	✓						✓						
12				✓				✓					
13	✓				✓								
14						✓							
Σ	7	2	3	1	5	2	1	1	1	1	1	1	1

Table 5.9: Navigator categories of participants, based on the number of actions from the two most used tools

applications and a further six users had at least one tool common between applications. This is an indication of users' preference for tools that are available across many applications, allowing them to become familiar, confident and accurate with their use.

5.8 Use of Other Navigation Tools

Vertical navigation makes up the majority of the two-dimensional scrolling actions observed during the study period. However, horizontal navigation, page layout and zooming actions are also important for controlling the document view. These are described in this section.

5.8.1 Horizontal Document Navigation

Horizontal navigation is supported by a much smaller set of tools than vertical navigation: the scrollbar, keyboard arrows, rate-based scrolling (Word only) and the hand-tool (Reader only).

Eight participants were observed to horizontally navigate the document, on a total of 114 occasions (see Table 5.10, lines 1 and 5–7). The scrollbar was the only tool used for horizontal scrolling (lines 5–7), with approximately one third of the horizontal scrolling actions occurring immediately following a zoom event (line 4). This is indicative of a mismatch between the horizontal position the user expected to see after a zoom action and the view actually provided. Horizontal scroll actions move, on average, 42% of the page width and take approximately 1.4 seconds (lines 8–9).

5.8.2 Page Layout Tools

Page layout tools modify the appearance of the document on-screen. Documents are opened with an initial layout that the user may modify during their interaction. The page layouts supported by Microsoft Word and Adobe Reader, and the naming conventions used by these applications, are described in Table 5.11.

Analysis	Mean	Med.	s.d.	Min	Max
<i>General</i>					
1 # participants who scrolled horizontally					8
2 # of docs. horizontally scrolled					36
3 # actions per document	3.2	2.0	3.7	1	19
4 % actions immediately after zooming					28.9%
<i>Division of actions</i>					
5 Scrollbar thumb					104
6 Scrollbar arrows					8
7 Scrollbar trough					2
<i>Action properties</i>					
8 Distance per action (% page width)	42%	36%	29%	1%	100%
9 Time period of actions (sec)	1.4	1.2	1.1	0.02	5.9
10 Velocity of actions (page widths/sec)	1.5	0.4	4.0	0.004	37.9

Table 5.10: Summary of horizontal scrolling actions

What layouts are used when documents are opened?

The application's layout of the document influences navigation—book layouts are navigated differently to a single continuous stream of pages. Microsoft Word's *print layout* and Adobe Reader's *continuous view* are the most commonly used views when a document is opened. Word's *print layout* accounts for 94% of document openings, the remainder be divided between *normal*, *reading layout* and *web layout*. Reader's continuous view accounts for 78% of openings, the remaining 22% used *Single Page* layout. These initial page layouts are selected from user preferences, the layout specified within the document or a default setting.

Do users change the document's layout?

A user can modify the page layout while interacting with the application. The vast majority (89%) of documents did not have their page layout changed. In Microsoft Word, thirteen participants changed the layout on 274 occasions in 203 different documents over the 120 period (14% of documents). In Adobe Reader, five participants changed the layout on 28 occasions in 19 different documents

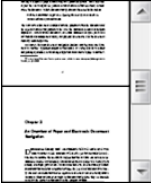


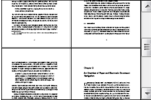

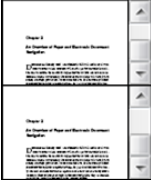
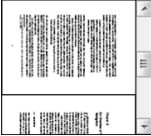
Layout/Tool	Appearance	Description
<i>Linear continuous</i> <ul style="list-style-type: none"> • Word: Normal, outline, print-layout and web-layout • Reader: Continuous view 		Pages flow smoothly from top to bottom.
<i>Linear discrete</i> <ul style="list-style-type: none"> • Word: Reading view • Reader: Single page view 		Only complete pages are shown, multiple partial pages cannot be viewed.
<i>Book view discrete</i> <ul style="list-style-type: none"> • Word: Reading view • Reader: Facing view 		Two pages are displayed side-by-side, as is seen in an open book.
<i>Book view continuous</i> <ul style="list-style-type: none"> • Reader: Continuous-facing view 		Similar to <i>book-view discrete</i> , except multiple, partial pages can be viewed.
<i>Multi-page view</i> <ul style="list-style-type: none"> • Supported by both applications when zoomed out in some views 		Many pages of the document are viewable at the same time.
<i>Split-views</i> <ul style="list-style-type: none"> • Word only 		Two different views, with independent navigation, onto the same document.
<i>Rotation</i> <ul style="list-style-type: none"> • Reader only 		Rotating the page by 90° from portrait to landscape view.

Table 5.11: Description and naming conventions of page layout tools available in Microsoft Word and Adobe Reader

View changed to:	Total # times	# docs	# participants	Mean pp.	Med.	s.d.	Min	Max
<i>Microsoft Word</i>								
Print preview	147	98	8	18.4	6	26.6	1	78
Print layout	44	38	11	4	3	4.4	1	16
Split view	42	32	7	6	3	8.2	1	24
Normal view	19	7	15	2.7	2	1.9	1	5
Outline view	9	4	7	2.3	2	1.3	1	4
Reading layout	7	4	7	1.8	1.5	1.0	1	3
Web layout	6	3	6	2	2	1	1	3
<i>Adobe Reader</i>								
Full screen	11	8	2	5.5	5.5	6.4	1	10
Rotate clockwise	6	2	2	3	3	1.4	2	4
Facing	4	2	2	2	2	1.4	1	3
Continuous	3	3	1					3
Continuous-facing	2	2	1					2
Single page	2	2	1					2

[†] pp. = per person, of the people who used this tool.

Table 5.12: Changes made to the page layout during document interaction (note, this data is reported as the total number of changes per user, rather than per document, as users rarely made these actions)

(1.4%). These page layout modifications are summarised in Table 5.12. The most common view change in Word was to *print preview* mode and in Reader it was to *full screen* mode.

5.8.3 Zoom Tools

Zoom tools allow the user to change the magnification of the document. Modifying the page layout or performing other document editing functions (such as adding a reviewing comment) may automatically adjust the zoom so that all of the required content is visible. However, both applications also contain several

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
<i>General</i>										
1 # of participants who changed zoom					10/13					14/14
2 % of docs. with zoom changed	16.0	17.3	11.3	1.4	40.1	19.5	21.6	10.9	0.9	40.6
<i>Document zoom over time (MoPM)</i>										
3 Mean zoom over time (%)	101.4	100.6	8.6	86.4	119.7	99.7	99.3	13.8	66.9	117.0
4 % of time with zoom below 100%	14.1	4.4	21.7	0.02	78.6	43.7	40.8	26.7	1.2	82.4
5 % of time with zoom above 100%	17.6	9.3	20.8	0.02	73.8	37.3	32.1	22.2	8.3	87.8

Table 5.13: Summary of zoom actions

	Word Zoom Tool	% of zoom actions	Reader Zoom Tool	% of zoom actions
1	Zoom combo-box	60.7% (s.d. 40.5%)	Zoom in/zoom out buttons	62.9% (s.d. 42.1%)
2	Ctrl-Mousewheel	31.6% (s.d. 38.9%)	Ctrl-Mousewheel	20.5% (s.d. 39.8%)
3	Keyboard shortcuts	7.8% (s.d. 18.8%)	Zoom in & zoom out tool	16.4% (s.d. 30.0%)
4			Zoom editable text box	0.2% (s.d. 0.9%)

Table 5.14: Summary of the use of zoom tools

tools for manually modifying the document's magnification. A summary of the zooming actions performed by the study group is provided in Table 5.13.

Ten of the Word users and all of the Reader users found the need to manually modify the zoom (line 1), in a small percent of the documents (line 2). The average zoom over time for both applications was close to 100% (line 3).

A summary of the tools that were used to modify the document's zoom is provided in Table 5.14. Most zoom tools provide discrete adjustment of the zoom level. The Ctrl-Mousewheel technique, while each wheel notch provides a discrete level, was often turned multiple times in quick succession to reach the desired magnification. This series of actions, within the normal two-second timeout, is described as a single zoom action. A similar principle applies to the keyboard shortcuts and zoom in/out interface buttons. Tools not included in this table (such as Reader's dynamic zoom) had no use observed during the study.

In Microsoft Word, the *zoom combo-box* was the preferred method for adjusting the zoom, accounting for 61% of actions. This is most likely due to the visibility this widget has on-screen. Both of the other techniques employed, Ctrl-Mousewheel and keyboard shortcuts, require the user to memorise the shortcut

mechanism. In Adobe Reader the *zoom-in* and *zoom-out* buttons were the most used zoom tool. Again, this is likely explained by the visibility of these buttons. Additionally, their use may also be attributed to their ‘obvious’ function—other tools, such as the *zoom in and out tool* first requires selection of a interface button and then depression within the document area.

Interestingly, in both applications, tools that would be classed as *expert tools*, Ctrl-Mousewheel and keyboard shortcuts, on average, accounted for less than a third of zooming actions, indicating a lack of knowledge of more advanced zooming features.

5.9 Navigation Patterns

Much of the data presented thus far has focused on individual navigation actions and their associated properties. This section takes a higher level view, looking at navigation patterns observed in the data. It covers the amount of time spent in document regions, the document’s position over its lifetime and within-document revisitation.

5.9.1 User Interest in Document Regions

The period of time spent in particular areas of a document provides an indication of interest in these regions. The properties of the scrollbar thumb are used to determine the length of time spent in different portions of the document.

The scrollbar thumb not only provides feedback on the current position in the document, the extent (length) of the thumb is also proportional to the percent of the document currently visible on-screen. A small thumb indicates a small percentage of the document is currently visible, while a large thumb indicates a large percentage is visible. AppMonitor records the size of the scrollbar trough, the thumb position and the thumb size. This data is used to calculate the currently visible *window* onto the document, along with the time spent in that window. The only caveat with this method is that in large documents, the extent of the scrollbar thumb is at the minimum and so may not provide an accurate measure of the window onto the document. Unfortunately, AppMonitor has no further method of determining this information, so the extent of the scrollbar thumb is used as the measure for all document windows.

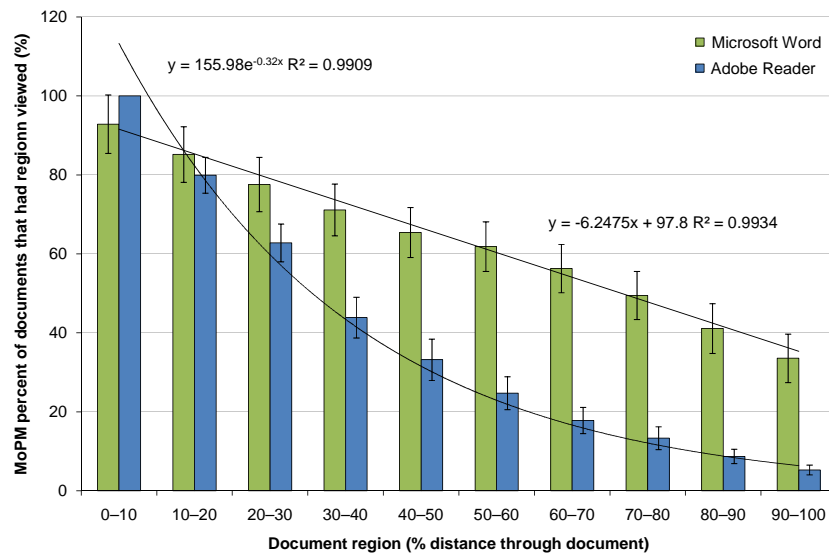


Figure 5.20: Percent of documents in which particular regions were viewed. Error bars show standard error.

Are documents viewed in their entirety?

Very few documents are viewed in their entirety. Figure 5.20 shows the percent of documents in which particular regions were viewed. For clarity, documents are divided into 10% chunks. On average, 64.6% (s.d. 13.6%) of a Word document is viewed and 34.0% (s.d. 11.1%) of an Adobe Reader document is viewed in each session (from opening of the document to closing it). Word documents have a linear decrease in the chance of a region being viewed the further it is toward the end of the document ($R^2 = 0.99$), while Reader documents follow an exponential decay ($R^2 = 0.99$). Documents opened in Reader are significantly more likely to have information at the start viewed than at the end.

How long do users spend in different document regions?

The percent of time spent in a document region decreases the further that region is towards the end of the document (see Figure 5.21). The largest amount of time is spent in the first 20% of the document, with a relatively constant amount spent in the middle sections. The final 90–100% chunk is provided the least amount of attention.

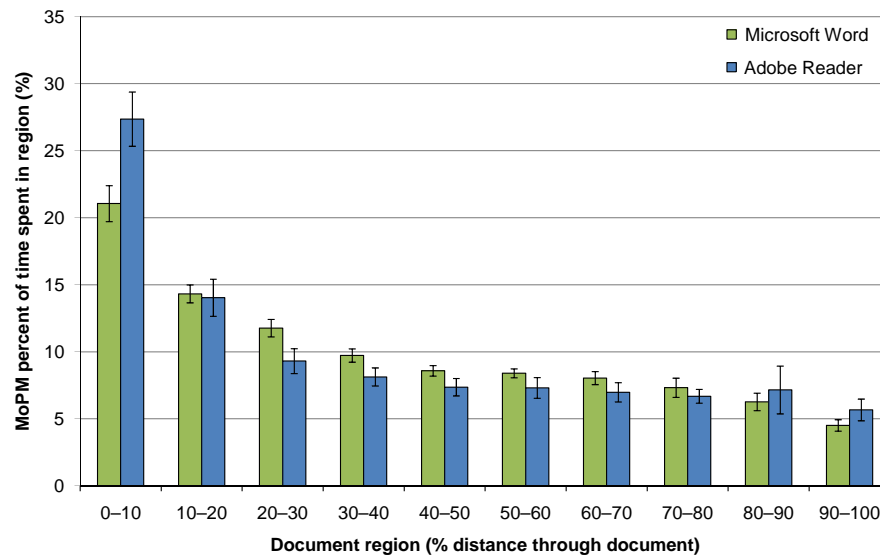


Figure 5.21: Mean percent of time spent in document regions. Error bars show standard error.

5.9.2 Activity Patterns

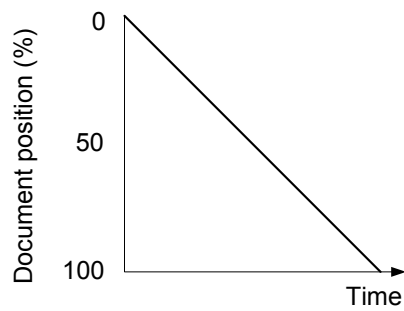
The position of the view onto a document, over its entire life, is rarely consistent between documents. However, some general patterns emerged from the analysis of the participants' data. These patterns provide an indication of the tasks users are conducting.

Can activities be identified from the logs?

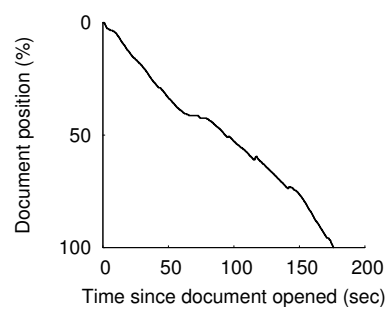
Four activity patterns were identified in the log files. Idealised figures of these patterns are shown on the left hand side of Figure 5.22 and examples from the collected data are shown on the right. Note, the graphs in this figure have the values on the y-axis reversed from that on traditional plots. This is to maintain the relationship between the scrollbar thumb position and the percent distance the user is through the document.

The activity patterns were termed *reading*, *studying/writing*, *skimming* and *cross-referencing*. These are described below:

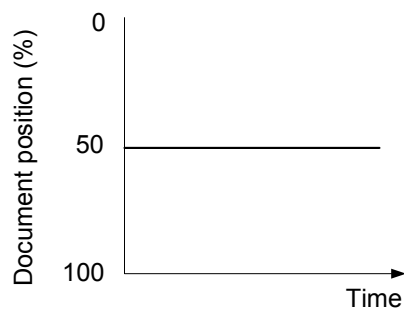
Reading: Figure 5.22a shows a linear increase in the position of the document



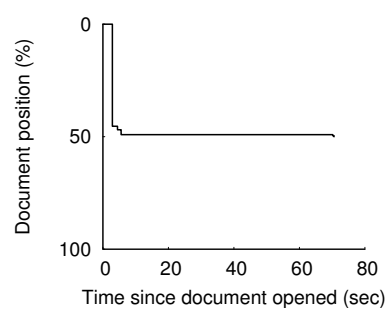
(a) Reading



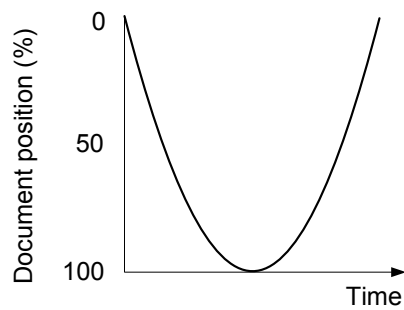
(b) Reading example (participant 10)



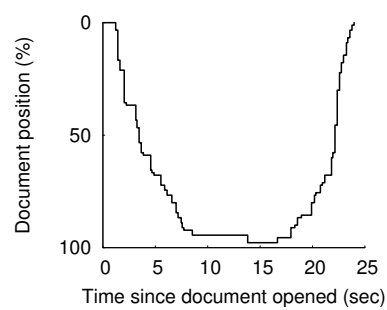
(c) Studying/writing



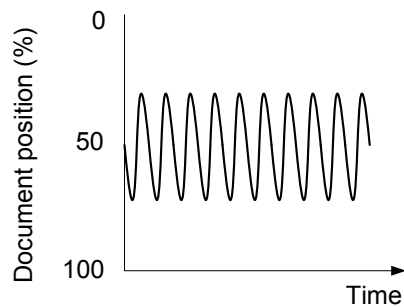
(d) Studying/writing example (participant 10)



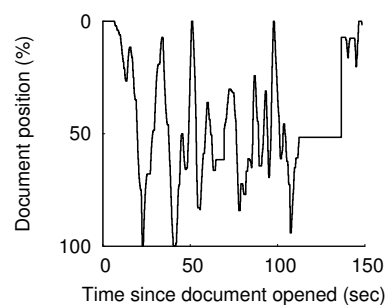
(e) Skimming



(f) Skimming example (participant 8)



(g) Cross-referencing



(h) Cross-referencing example (participant 1)

Figure 5.22: Generalised and example document positions over time. Note the reversed vertical axis to align with scrollbar thumb positions.

over time (recall that this is an increase, not a decrease, due to the axis scale). In this case, the user is simply moving from the beginning of the document to the end, without any back-tracking, a pattern that is likely when reading a document.

Studying/writing: An example of a *studying/writing* pattern is shown in Figure 5.22c. Documents that fall into this category are quickly moved to a particular position, with the remainder of the document's life then spent at or near that location.

Skimming: Figure 5.22e illustrates a curve pattern of navigation—users move through the document and then return it to the beginning, an action indicative of quick document skimming.

Cross-referencing: Figure 5.22g shows document *cross-referencing*. This occurs when the user moves backwards and forwards through a region of the document, many times.

Documents that do not clearly fit in any of these categories or documents that show elements of several categories are placed into an *other* group.

What is the distribution of these activities?

Microsoft Word documents most commonly showed the studying/writing pattern, while Reader documents were most commonly used for reading (see Figure 5.23). The *studying/writing* pattern in Word is likely due to users' spending their time editing a specific region of the document. Word documents more commonly show cross-referencing behaviour (users moving the viewport backwards and forwards over an area of the document) than Reader documents. This is most likely due to the navigational requirements of editing actions (writing and re-writing often require returning to previously written material). As expected, Adobe Reader documents were predominately used for reading activities.

5.9.3 Within-Document Revisitation

Within-document revisitation is the act of returning to a previously viewed position within the document.

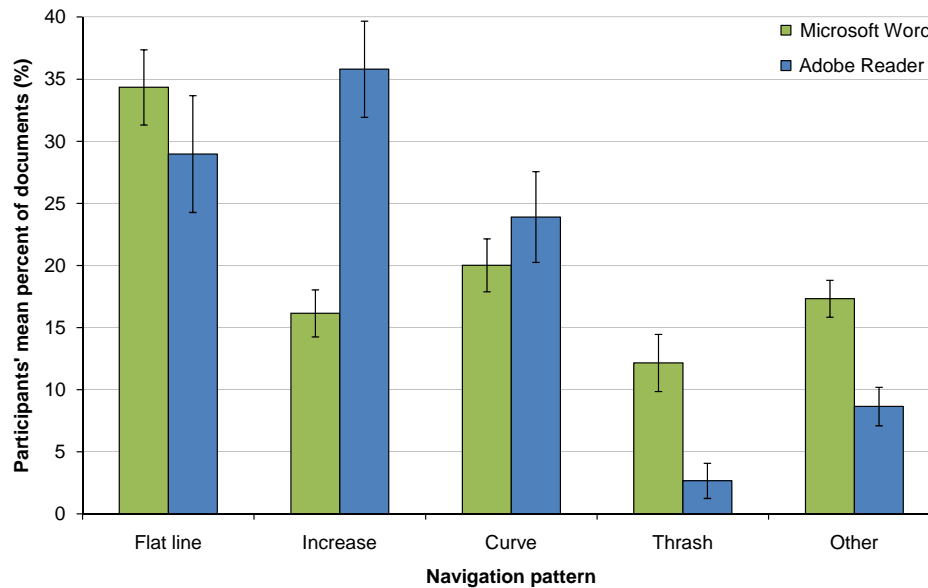


Figure 5.23: Distribution of navigation patterns over the documents' lifetimes. Error bars show standard error.

Defining Visits and Revisits

When navigating within documents, people frequently stop over a region without any intention to do so. This is caused by mechanical demands such as clutching the mouse or by cognitive and perceptual issues such as determining whether the document is correctly positioned. As with the previous analyses reported in this chapter, a timeout of two seconds is used to define an intentional pause. This is used to prune visits of shorter duration.

Further, when revisiting document locations, people are unlikely to arrive at precisely the same viewport as the one previously seen. A revisit is therefore defined as returning to anywhere within the bounds of an earlier visit and remaining there for more than two seconds. Furthermore, revisits are only logged when the person visits another position outside the current view prior to returning. This condition ensures that actions such as slowly advancing line by line are not logged as revisits.

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
1 Total documents					2271					1706
2 Total occurrences of revisitation					7622					1222
3 % documents with interaction	63.4	63.8	9.8	51.1	89.2	83.9	82.3	9.4	69.9	100
4 % documents with revisits	33.2	30.8	12.9	13.0	61.5	31.7	31.7	13.2	5.9	53.8
5 Mean max. daily revisits	203.3	254	104.3	4	332	32.3	26.5	28.3	1	118

Table 5.15: Summary of within-document revisitation in Microsoft Word and Adobe Reader from the longitudinal study

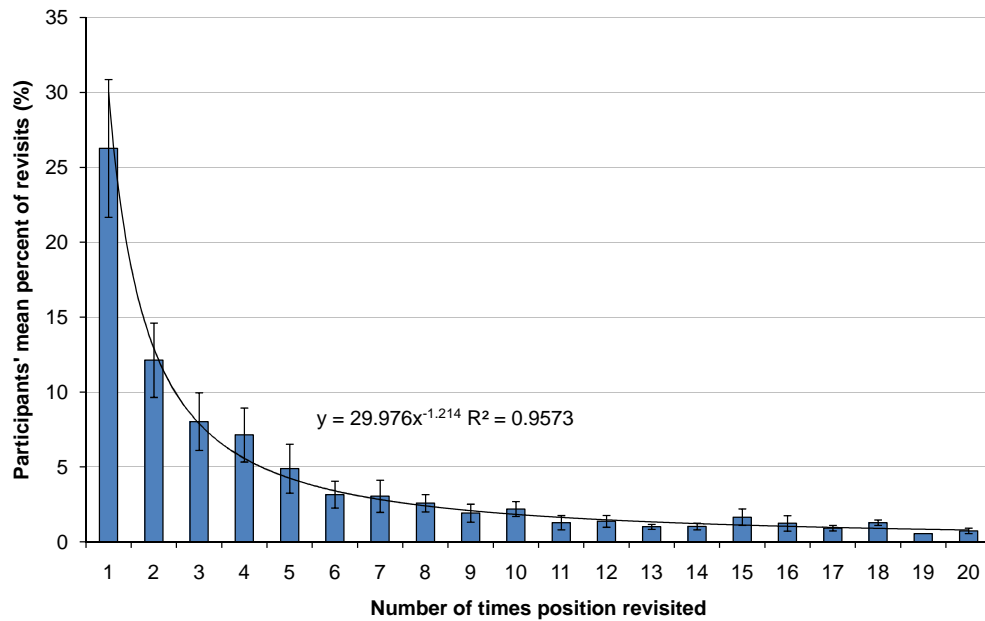
Patterns of Revisitation

Revisitation is a commonly performed action. People frequently revisit document regions (Table 5.15, line 2), in a large portion of the documents with interaction (lines 4 and 5). On some days, this can occur several hundred times (line 6). Positions are revisited more often in Word than they are in Reader, as illustrated in Figure 5.24. The editing actions performed by Word users account for this greater revisitation rate—writing and re-writing often requires the user to return to view previously created content.

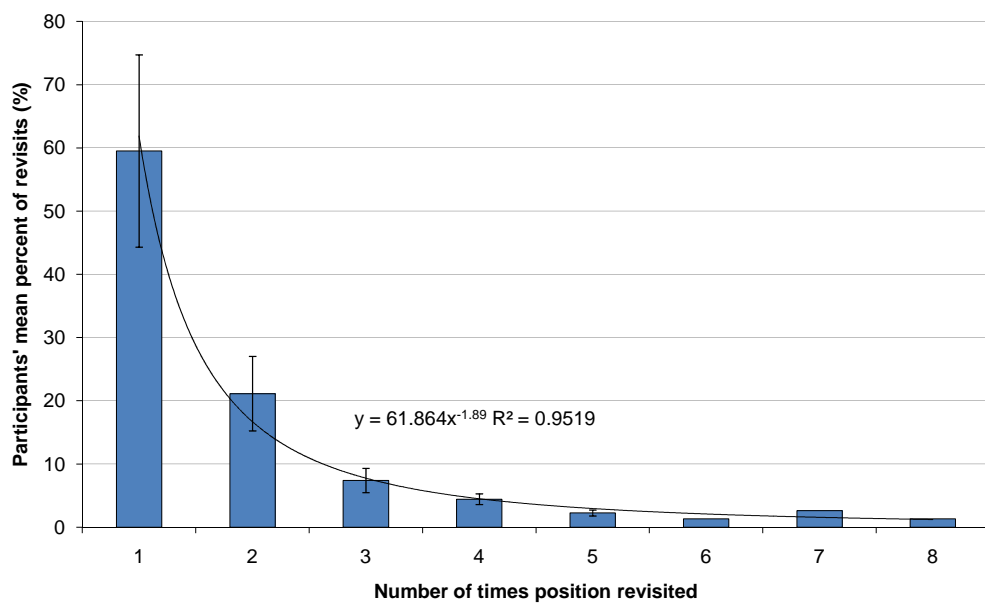
This leaves the potential for efficient tools to significantly improve user satisfaction, by both decreasing reacquisition time and by reducing user frustration from continually searching for document regions that they have previously viewed.

Use of Revisitation Tools

Both of the applications under study provide tools that aid the user in returning to previously viewed document positions. These tools include bookmarking functions, split-views and view navigation. As reported earlier, none of the participants used Word’s bookmarking functions, while Reader’s bookmarks were used 65 times during the study period. Word’s *split window* feature allows users to simultaneously view more than one document region, eliminating the need for repetitive scrolling between two positions. Seven of the fourteen participants used split windows at least once, with a total of 42 invocations. One participant accounted for more than half (24) of these events.



(a) Microsoft Word



(b) Adobe Reader

Figure 5.24: The number of times positions are revisited within a document

Adobe Reader's view navigation buttons allow users to revisit regions of the document they have previously seen. Reader's algorithm for what constitutes a view or when a view is added to the history is not publicly documented. One participant used the *previous view* navigation tool on two separate occasions (in two separate documents). In each instance, seven clicks of the previous view interface button were recorded, so it can be assumed these were intentional actions and not accidental 'slips'. No further use of the *previous view* or any use of *next view* were recorded.

Overall, within-document revisitation is a commonly performed action, but the tools that can aid this task are rarely utilised.

5.10 Further Observations

Understanding users' document navigation actions was the primary focus of this longitudinal study. However, several other useful observations of interaction with the applications were observed. These include: the use of reviewing tools, document and window management, and the use of menu items, toolbar buttons and keyboard shortcuts. While some of these actions are not specifically related to document navigation, the findings are potentially useful to other researchers and so are reported in this section.

5.10.1 Use of Reviewing Tools

Microsoft Word's reviewing tools provide functionality to support the collaborative writing of documents. These features allow users to 'track' the changes they are making (additions, deletions, and modifications of content and presentation), add and remove comments on regions of the document, and to accept or reject the changes made by other authors. Clearly, the use of these functions indicates the user is reviewing or editing work that is the contribution of at least one further author. The observed use of such tools is summarised in Table 5.16.

Eight of the thirteen participants took part in collaborative authoring (note that these actions could be made for purely personal use, but this is unlikely), although in only a small percent of documents (Table 5.16, line 2). These actions are likely to take place early in the document's life (line 4).

Analysis	Mean	Med.	s.d.	Min	Max
1 # participants who used reviewing functions					8/13
2 % docs. with reviewing actions	12.1	11.4	8.1	1.4	26.5
3 # reviewing actions per doc.	5.5	4.2	4.7	2	16.7
4 Mean % time actions are from opening doc.	35.3	35.6	13.3	8.5	52.4

Table 5.16: Summary of the use of reviewing tools in Microsoft Word

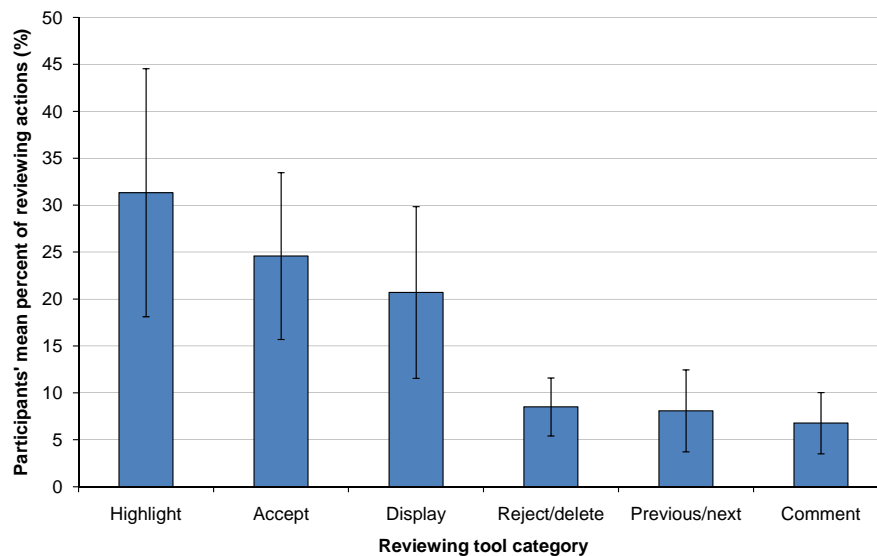


Figure 5.25: Distribution of use of reviewing tools. Error bars show standard error.

The *highlight* and *accept* actions were the most commonly used, accounting for an average of 31% and 25% of actions respectively (see Figure 5.25). The large standard error bars are indicative of the diverse use of these tools by participants.

5.10.2 Document and Window Management Activities

Document management activities primarily involve opening and closing documents, while *window management* activities encompass the actions of minimising, maximising, moving and resizing the window, and bringing it to the foreground. Window management activities allow the user to make use of their available screen real-estate and to work in multiple documents or applications simultaneously.

Analysis	Microsoft Word	Adobe Reader
<i>Opening Documents</i>		
1 # from <i>File</i> → <i>Open</i> menu item	13	1
2 # from <i>open</i> toolbar button	116	16
3 # from <i>Ctrl-O</i> keyboard shortcut	160	0
4 # from recent files in <i>file</i> menu	66	2
5 % docs.	15.6%	1.1%
<i>Closing Documents</i>		
6 # from <i>Close</i> push button	1134	739
7 # from <i>File</i> → <i>Close</i>	2	0
8 # from <i>File</i> → <i>Exit</i>	3	0
9 # from <i>Ctrl-W</i> , close shortcut	106	8
10 # from <i>Ctrl-Q</i> , exit shortcut (Adobe)	–	0
11 # from <i>Alt-F4</i> , exit	4	0
12 # from <i>System</i> → <i>Close</i>	50	13
13 % of docs.	57.2%	44.5%

Table 5.17: Summary of document management activities

How are documents opened?

The majority of documents are opened from outside the studied applications. Only 15.6% of Word documents and 1.1% of Reader documents were opened from within the application (Table 5.17, lines 1–5). This indicates regular use of actions such as double-clicking a file in a Windows Explorer window or directly opening downloaded documents from a web-browser.




One explanation of this behaviour may lie with the lack of context traditional open mechanisms provide—anecdotally, users quickly get frustrated with open dialog boxes that do not open in the “correct” directory for the required file.

Within-application history mechanisms, such as the recent files list, were also poorly utilised, despite 66% of Word 40% of Reader documents attributed to re-openings of previously viewed documents. This behaviour is reflective of that observed in web-browsers, where users rarely employ bookmarking mechanisms for revisiting pages (for example, Tauscher and Greenberg found bookmarks accounted for only 3% of revisits [214, pg. 110]).

How are documents closed?

Documents are almost always closed using the *close* button (✕) in the top right-hand corner of their screen (Table 5.17, line 5). This is more convenient than other methods, such as selecting *close* or *exit* from the *file* menu and is comparable in speed, and less memory effort than using the shortcut keys (for example, Alt-F4). Note that these values do not account for all open documents—pressing the close button in reader can close multiple documents simultaneously. Other methods for closing a document, such as taskbar context menus or letting the operating system close the document as it shuts down, are not recorded by AppMonitor.

Window Management

Window management activities manipulate the size, position and visibility of the application (and in this case the document) on-screen. AppMonitor can record six such actions in this category: moving the window (by dragging the title bar), minimising the window to the task bar (using the  button), restoring the window (unmaximising it to the previous size and position, using the  button), maximising the window (to take all available screen space, using the  button), manually resizing the window (by dragging the corners or sides of the window) and bringing the window to the foreground (by clicking within the window or using the Alt-Tab window switching mechanism). A summary results of these actions is provided in Table 5.18.

Maximising, restoring, moving and manually resizing windows occurs very little (Table 5.18, lines 13, 16 and 20). Generally, there are not other documents from the same application open when these operations occur, an indication that these functions are *not* used to view multiple documents from the same application simultaneously. Instead, it is likely these actions took place to comfortably fit the contents on-screen. The Windows operating system will generally ‘remember’ the last size and position at which an application was open and if this position was comfortable, the user is unlikely to need to regularly perform move or resize operations.

Most users minimised around a quarter of their documents and those that were minimised, were minimised on average twice (lines 5–7). Once a document was minimised, it generally spent a long time in that state, with an average of six

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
<i>Moving the window</i>										
1 # participants who moved					11/13					10/14
2 % docs. moved	15.2	7.4	15.1	1.9	43.3	11.5	6.6	10.5	1.8	28.5
3 # moves per doc. with ≥ 1 move	2.8	2.6	1.6	1	5.6	1.3	1.3	0.2	1	1.6
4 # other docs. open when moved	0.5	0.7	0.5	0.2	1.4	0.9	0.2	1.6	0	4.7
<i>Minimising the window</i>										
5 # participants who minimised					10/13					13/14
6 % of docs. minimised	24.7	28.2	15.2	4.7	48.5	25.8	22.5	18.7	3.0	71.4
7 # minimises per doc. with ≥ 1 minimise	2.0	1.7	0.8	1.3	3.3	1.9	2.0	0.4	1	2.5
8 # other docs. open when minimised	1.0	0.7	1.1	0	3.8	0.6	0.2	1.0	0	3.7
9 Mean time spent minimised, per action (min)	369.3	405.1	286.5	8.0	829.6	115.6	73.9	144.9	0.03	432.7
10 Mean time spent minimised, per doc. that was minimised (min)	724.8	498.3	717.3	13.8	2371.9	220.3	149.8	320.3	0.03	1009.7
11 % time spent minimised, per doc. that was minimised	36.7	34.1	20.1	5.2	66.3	32.7	35.0	16.2	2.2	56.9
<i>Maximising the window</i>										
12 # participants who maximised					10/13					11/14
13 % of docs maximised	5.7	3.1	7.2	0.5	22.3	13.6	3.6	19.8	1.0	69.6
14 # maximises per doc. with ≥ 1 maximise	1.1	1.0	0.1	1.0	1.4	1.1	1.0	0.2	1.0	1.5
<i>Restoring the window</i>										
15 # participants who restored					9/13					7/14
16 % docs. restored	3.8	3.7	3.4	0.5	12.1	3.1	2.2	2.8	1.0	9.3
17 # restores per doc. with ≥ 1 restores	1.1	1.0	0.2	1	1.4	1.0	1.0	0.1	1.0	1.2
18 # other docs. open when restored	0.7	0.3	0.9	0	2.5	1.1	0.4	1.5	0	3.5
<i>Manually resizing the window</i>										
19 # participants who resized					10/13					7/14
20 % docs. resized	6.4	4.0	6.0	2.3	20.6	12.1	7.1	14.6	0.9	43.8
21 Resizes per doc. with ≥ 1 resize	1.6	1.5	0.6	1	2.5	1.4	1.4	0.6	1	2.8
22 # other docs. open when resized	0.6	0.5	0.6	0	1.9	0.1	0	0.2	0	0.4
<i>Bringing the window to the foreground</i>										
23 # participants who brought to foreground					13/13					13/14
24 % of docs. brought to foreground	56.2	57.6	33.2	12.0	100.0	26.3	28.1	17.0	4.0	67.9
25 Foreground actions per doc. with ≥ 1 foreground action	4.2	4.0	2.1	1.6	8.9	2.2	2.2	1.1	1.0	4.2
26 # other docs. open when brought to foreground	1.0	0.5	1.1	0	3.9	0.4	0.2	0.7	0	2.4

Table 5.18: Summary of window management operations

hours for Word documents and nearly two hours for Reader documents (line 9). However, large standard deviations in this time indicates a wide variety of periods spent minimised. Overall, documents that were minimised spent 37% for Word and 33% for Reader of their time in that state (line 11).

All of the Word users and all but one of the Reader users brought the document's window to the foreground (line 23). This action occurs when the user has moved away from the document (for example to check their email or to do a web-search) and later returns to interact with this document by giving it focus. On average this occurred in 56% of Word documents and 26% of Reader documents (line 24) with a high likelihood of it occurring several times (line 25).

Other window management commands, such as side-by-side comparisons, cascade windows and selecting a document to switch to by using the *Window* menu were rarely used.

Many other window management activities occur outside individual applications and are controlled by the operating system. Studies such as Hutchings et al.'s [106] provide a more complete, in-depth analysis of this area.

5.10.3 Use of Menu Items, Toolbar Buttons and Keyboard Shortcuts

Menu items, toolbar buttons and keyboard shortcuts are used to select application functionality. Some of these functions support document navigation activities (such as changing the page layout), others trigger application-specific features.

Menu Selections

Menus, like scrollbars, are a cornerstone of virtually all GUI toolkits, with almost universal use in window-based applications. They traditionally provide access to all of the system's functionality, for example, Microsoft Word 2003 has over 250 menu items, which may lead to issues when searching or selecting items. Since this study was run, the next version of the office suite, Microsoft Office 2007, has been released, in which menus are replaced by a tabbed-toolbar, known as the *ribbon* (see Figure 5.26).

A total of 1813 selections were made by all users from Word menus over the 120 day period. Figure 5.27 illustrates the mean distribution of these menu selections. Word's most used item, *File* → *Print* accounted for 284 selections or

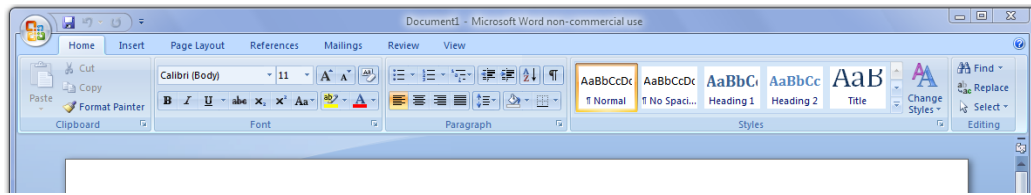


Figure 5.26: Microsoft Office Word 2007’s Ribbon—a replacement for traditional menus

Analysis	Microsoft Word					Adobe Reader				
	Mean	Med.	s.d.	Min	Max	Mean	Med.	s.d.	Min	Max
1 # participants who made menu selections					11/13					12/14
2 % menu selections that were direct	75.8	81.2	14.5	46	91.5	78.2	91.1	32.0	12.5	100
3 # participants who made indirect menu selections					11/11					7/12
4 # additional selections [†]	1.7	1.6	0.5	0.9	2.8	1.8	1.6	2.1	0	6.3
5 # additional top level menu selections [†]	1.2	1.3	0.6	0.2	2.5	0.9	0.6	1.0	0	2.9

[†] per indirect menu selection

Table 5.19: Summary of menu hunting

an average of 17.4% of user actions. This is followed by *File* → *Save As...* at 11.2% and *Format* → *Paragraph...* at 7.2%. Most menus had one item that was used more often than all other items. Cascades with no selections are not shown.

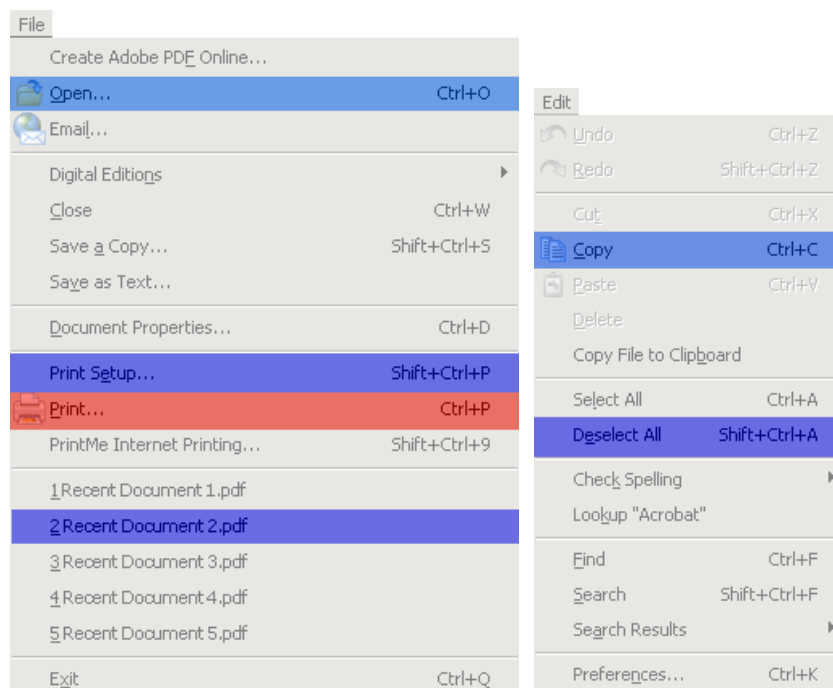
Adobe Reader had significantly less selections, with a total of 147, less than 10% of the number made with Microsoft Word. Figure 5.28 illustrates the mean distribution of these selections. As with Word, the most used item was *File* → *Print*, contributing 66 of these actions (or an average of 39.5% of user actions). This is followed by *Window* → *Full Screen View* at 16.7% and switching to the first open document, 11.9%. Again, un-used menus and cascades are omitted from the visualisation.

Menu Hunting

Menu hunting, the act of posting an incorrect top-level menu (for example *File* or *Edit*) or opening an incorrect cascading sub-menu, was regularly observed when menu selections were underway. AppMonitor was configured *not* to record all

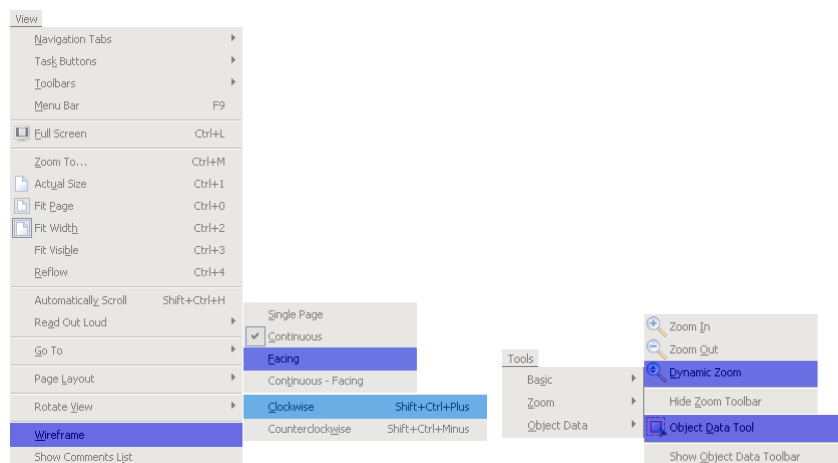


Figure 5.27: Participants' mean distribution of the use of Microsoft Word's menus. Colours represent distribution percents, as illustrated by the usage scale in sub-figure (j).



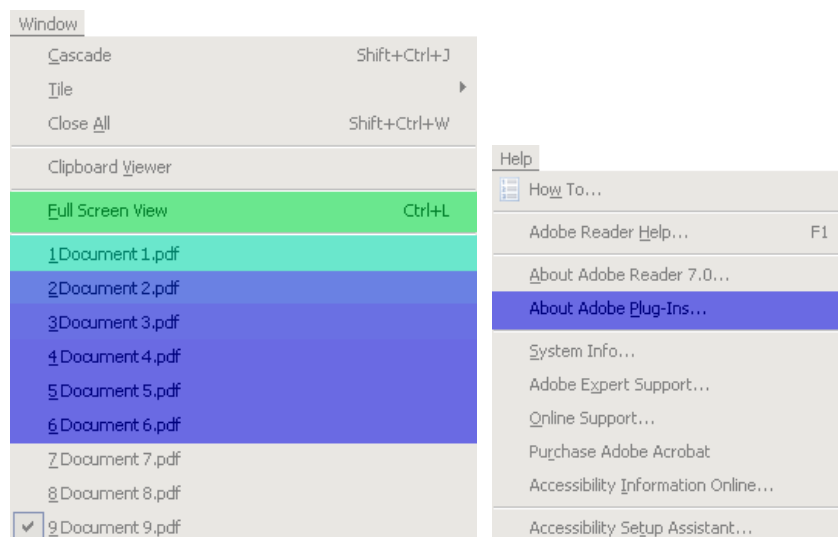
(a) File menu

(b) Edit menu



(c) View menu

(d) Tools menu



(e) Window menu

(f) Help menu

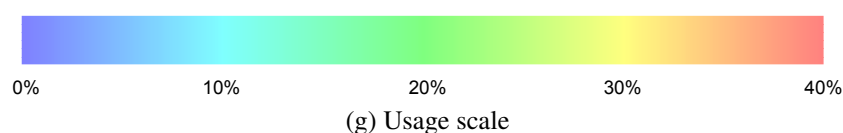


Figure 5.28: Participants' mean distribution of the use of Adobe Reader's menus. Colours represent distribution percents, as illustrated by the usage scale in sub-figure (g).

mouse movement (reporting the object the cursor was currently over), however, it does record whenever a menu or cascade is *posted* (opened). All of the Word participants who selected from menus made at least one hunting action (Table 5.19, lines 1 and 3). Between 22% and 25% of menu selections contain hunting actions (line 2), with 1.7–1.8 *additional* selections made per hunting action (line 4). Over half of these actions arise from selecting an incorrect top level menu (line 5). Researchers have previously focused on increasing the speed of menu selections *within* the menu (for example, Sears and Shneiderman [198]), but may also want to consider mechanisms for aiding correct top level menu selection.

Context Menus

Context menus are triggered by right-clicking within an application. These provide quick access to context-sensitive and/or commonly used functions. Users of Microsoft Word are more likely to use context menus than their counterparts using Adobe Reader. Word users opened context menus a total of 1640 times in 303 documents; in contrast, Reader users only triggered context menus a total of 57 times.

The most commonly performed action after opening a context menu was to cancel it, by clicking outside of the menu area (see Figure 5.29). This accounted for an average of 26.6% of actions. Such a large number of incorrect openings of the context menu is potentially due to motor-slips or because the menu did not contain the required menu item. Spelling and grammar functions, including correcting mis-spelled words and selecting synonyms made up a further 26.5% of selections. The large standard error for this category is explained by individual use of context menus for this group of actions: this accounted for 100% of one participant's actions and only 2% of another's actions. Cut/copy/paste actions, table and picture formatting, reviewing and general formatting made up majority of the remaining selections.

In Reader, 68% of the context menu actions were immediately cancelled (39 out of a total of 57 selections). The remainder were evenly split between selecting the *copy* menu item (five participants) and selecting some other menu item (two participants).

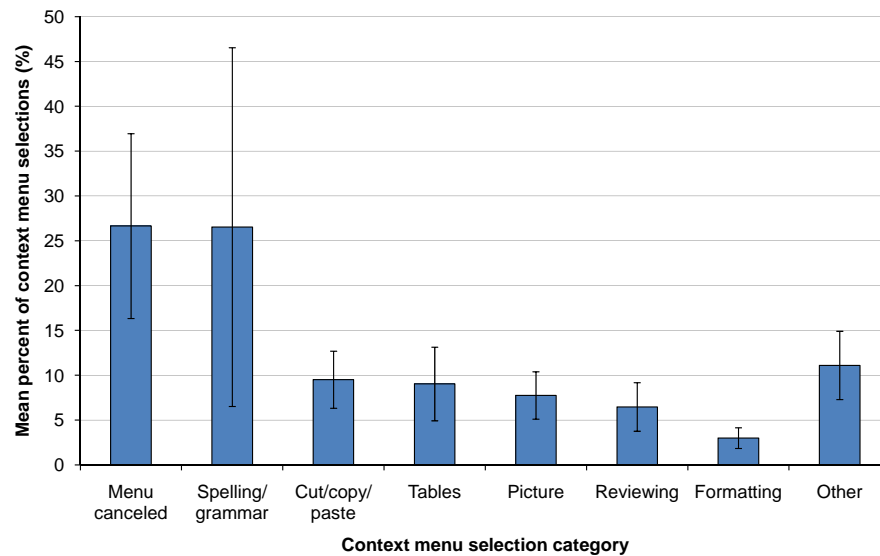


Figure 5.29: Context menu selections for Microsoft Word. Error bars indicate standard error.

Use of Toolbar Buttons

Toolbar buttons provide quick access to commonly used application functions. Microsoft Word users selected a total of 2787 toolbar buttons and Reader users 951. Figure 5.30 provides a visualisation that summaries this use for the two applications under study. In each application, it is clear that a single toolbar button dominates over all others. In Microsoft Word (Figure 5.30a) the *save* button has the greatest use, with an average of 23% of actions, with the remaining buttons all falling in the 0–9% range. In Adobe Reader (Figure 5.30b) the *zoom in* button has the greatest use, with 32%; however, this is followed more closely by the *zoom out* button with 21%. The *text select* and *next page* buttons also feature more highly than other actions.

Perhaps surprisingly, but also illustrative of the expert level of our users, none of the Microsoft Word users pressed the cut, copy or paste buttons during the 120 day study period. Instead, these functions were accessed through the menu items and keyboard shortcuts (see the previous and following sections).

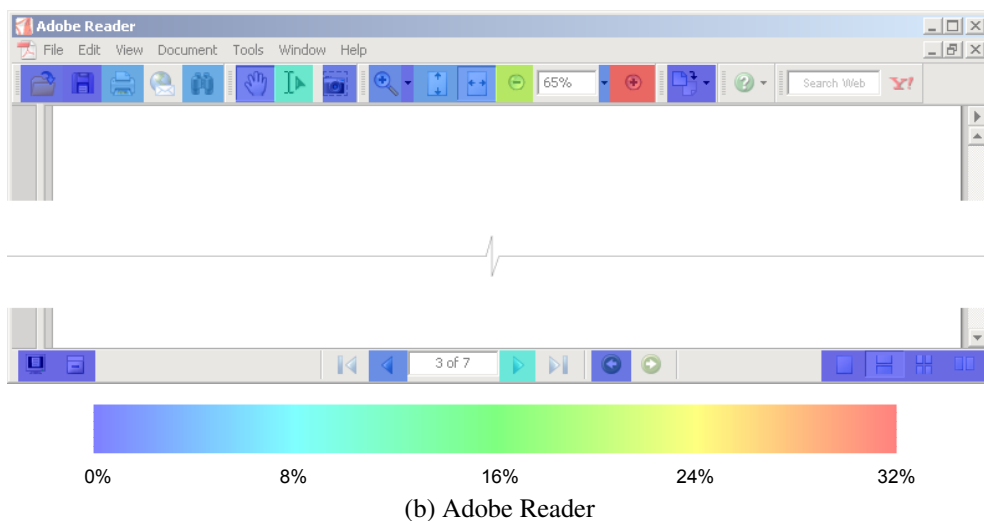
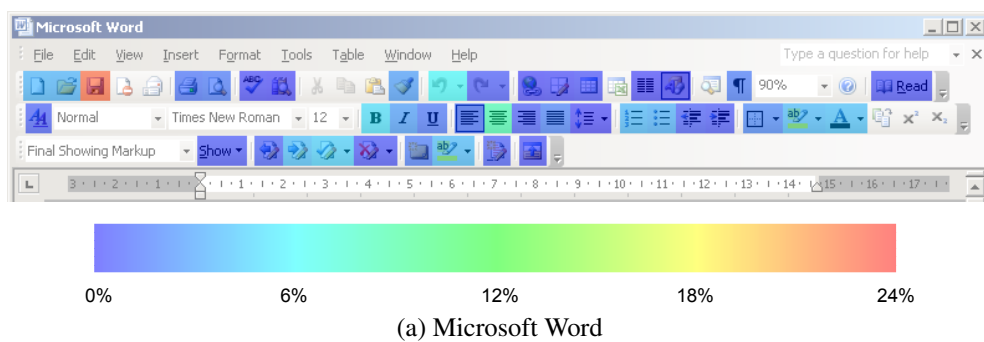


Figure 5.30: Distribution of toolbar button use. The visualisation shows the participants' mean percent of actions for each button.

Keyboard Shortcuts

AppMonitor was configured to only record shortcut key presses during the longitudinal study. Shortcut keys provide, as their name suggests, a quick method for accessing commonly used functions. AppMonitor also classes the use of the function keys, the arrow keys, the paging keys, tab, home and end keys as shortcuts.

When keys are held depressed, auto-repeat functionality engages, rapidly sending *key pressed* messages to the focused application. Auto-repeat is useful for some functions (such as holding down the arrow keys to slowly move through the document), but not so for other functions, such as changing the font style to bold. AppMonitor does not distinguish auto-repeat key-presses from standard presses, as the delay and repetitions per second are user configurable (although default values are approximately 500 ms and 33 repeats/sec respectively). Post-study analysis could attempt to separate these two types of key presses, however without wider knowledge of the user's system configuration, this analysis could be inaccurate.

The rankings of individual key presses are reported by the frequency of key use. This frequency is calculated by taking the mean of the participants' frequency of use, which is calculated as the number times a key is pressed, divided by the number of documents with interaction, or:

$$f_k = \frac{1}{n_p} \sum_{i=0}^{n_p} \left(\frac{n_{ki}}{n_{di}} \right) \quad (5.3)$$

where:

- f_k = frequency of use of shortcut key, k
- n_p = number of participants
- n_{ki} = number of times participant i used key k
- n_{di} = number of documents with interaction for participant i

Microsoft Word users recorded an average of 5947 shortcut key presses and Adobe Reader users an average of 529 over the study period. The eight most regularly used shortcut keys in Microsoft Word were for navigation purposes, the top four

Key	Use frequency (uses/doc)	# who used (/12)
1 Down arrow	9.0	12
2 Up arrow	5.2	12
3 Left arrow	5.0	12
4 Right arrow	4.5	12
5 Ctrl-right arrow	2.7	6
6 Page down	2.6	7
7 Page up	2.2	6
8 Ctrl-left arrow	2.1	6
9 Ctrl-S	2.1	11
10 End	2.1	12
11 Ctrl-V	1.8	12
12 Ctrl-Z	1.6	12
13 Shift-Ctrl-right arrow	1.5	5
14 Shift-Ctrl-left arrow	1.3	5
15 Shift-left arrow	0.9	12

Table 5.20: The fifteen most frequently used shortcut keys in Microsoft Word (note one participant is excluded from this analysis, as they recorded only 21 shortcut key presses, compared to the average of 5947)

accounted for by the four directional arrow keys. The shortcut keys for save, paste and undo were the only non-navigation keys that appear in the top fifteen, as shown in Table 5.20.

A similar pattern, of navigation keys dominating shortcut presses, is also observed in Adobe Reader (see Table 5.21). While Word shortcut keys are dominated by the small movement arrow keys, Reader's paging keys, especially *page down* are dominant over all other presses. Small distance movement with the down arrow scores third. The only non-navigation shortcut to appear in the top 10 most frequently used keys is the shortcut for copy, `Ctrl-c`.

5.11 Discussion

This chapter has presented an empirical characterisation of the document navigation habits of 14 participants over 120 days. The implications of this characterisation on navigation tool designers and system designers are first discussed. The

Key	Use frequency (uses/doc)	# who used (/10)
1 Page down	6.9	6
2 Page up	2.3	6
3 Down arrow	1.1	8
4 Space [†]	1.0	4
5 Left arrow [†]	0.7	4
6 Enter [†]	0.3	8
7 Up arrow	0.3	8
8 Right arrow [†]	0.2	4
9 Ctrl-Page down [‡]	0.1	1
10 Ctrl-C	0.04	6

[†] Recall that these keys may act as paging keys

[‡] This differs from *page down* by guaranteeing to move to the top of the next page

Table 5.21: The ten most frequently used shortcut keys in Adobe Reader. Note, four subjects had less than 10 key presses so were excluded from this analysis.

limitations of this study, areas for future work and recommendations for other researchers are described later in this section.

5.11.1 Implications for Navigation Tool Designers

Navigation tool designers can use this characterisation as motivation for the development of modifications to current tools or the design of innovative new mechanisms for moving within a document. Each piece of usage data will inform designers on possible areas for improvement. These improvements may range from simple interface adjustments, to complex interaction mechanisms, to new designs for input devices. To illustrate, three possible improvements that arise directly from the data presented in this chapter are described below:

Simple interface adjustments: Bookmarks were used significantly more often when their containing panel was visible as the document was opened. Configuring the interface to *always* show this panel (unless overridden by the user) would encourage the use of this tool for efficiently moving to document positions. Ensuring documents contain appropriate bookmarks (or

systems for automatic bookmarking) are outside the domain of human-interface designers, but should also be considered.

This technique applies to many aspects of navigation—zoom tools that had prominent positions on the interface were used in preference to advanced shortcut actions, as they did not require the user to memorise and recall shortcut triggers.

Tool additions: Within-document revisitation is a commonly performed task, yet tools to support this task (such as Word’s bookmarks and Reader’s previous/next view) were rarely, if ever, used. Users would benefit from a tool that visualises their visited document positions and aids them to quickly return to these positions. This could potentially be as an augmentation of the scrollbar.

Complex interaction mechanisms: Users of Adobe Reader’s hand-tool had a mean drag distance close to the maximum possible on a standard screen at 100% zoom. Users showed a desire to use such a technique, as it was the fourth most used Reader tool. However, this result shows users are attempting to move larger distances than those for which it was designed. Developers might consider adding finely tuned acceleration or flicking gestures to this tool, to increase its efficiency and navigation range.

Input device improvements: The mouse was heavily used in most electronic document navigation—the two most used tools, the mousewheel and dragging the scrollbar thumb, both require mouse interaction. However, this input requires the user to move their hand from the keyboard to the mouse, especially when editing a document. Interaction designers might consider improved methods for combining the keyboard and mouse to eliminate this physical movement time (several commercial products already do this, but they are not in widespread use). Interaction with the mouse could also be eased, by providing *auto-repeat* functionality on paging buttons or alternate methods for triggering navigation methods such as rate-based scrolling.

5.11.2 *Implications for System Designers*

One of the most significant findings of this study is the observation that participants primarily using a small sub-set of the available tools. Further, this sub-set is generally application independent. For example, when users were categorised into stereotypical navigators, five participants fell into the same categories across applications and six users had at least one of their two most frequently used tools in common.

This result impacts on application designers in two ways. First, *not* including mechanisms in the user's subset could severely hamper their use with the designer's new system. While users will learn new tools if they are forced to, it is advisable to not make them conquer the learning curve of another tool in order to use the application. Second, the incorporation of new navigation tools should be encouraged, especially those that increase the user's efficiency, however, they *should not* be included at the expense of tools with which users are familiar. Further, designers should not have expectations that their new tools will be regularly used—this study showed that efficient navigation mechanisms, such as the book-marks were rarely employed.

5.11.3 *Limitations*

This chapter characterised the document navigation habits of the 14 participants in our study. All participants were members of Computer Science departments and classified themselves as *advanced* or *expert* users. There is no reason to believe these results do not generalise to any *expert* user. A larger, more broad field study, with participants who do not classify themselves as experts would be required to determine whether navigation patterns change as proficiency increases.

One of the primary disadvantages of studies that utilise logging techniques, is their inability to provide contextual information. The data cannot provide information on the task the user was trying to achieve or why a particular tool was used for a particular task. The following chapter investigates these issues, by conducting two task-centric studies that ask users to perform *specific* navigation tasks. The first constrains participants to 'ideal use' scenarios for advanced navigation tools to test their knowledge of these techniques. The second uses interactive sessions to understand *why* particular tools are employed for particular tasks.

5.11.4 Future Areas for Similar Work

Characterisations of this kind would also be informative for other domains, for example mobile computing, spreadsheet applications and code editing. Mobile devices now come equipped with fully functional web-browsers, PDF viewers and word processing applications. All of these applications must operate on miniature screens, with alternative forms of input (including a stylus or simply a finger on a touchscreen). While there is much work going into providing new navigation mechanisms for this domain (such as using inbuilt cameras and physical device movements), a study that describes how users currently navigate on small screens would significantly benefit developers. Similarly for spread-sheeting and code-editing. Both of these activities are performed by millions of people world-wide, yet, like document navigation, the primary navigation tool remains the scrollbar. Cross-domain comparisons of navigation would also reveal interesting similarities or differences based on the task at hand and the content displayed.

5.11.5 Recommendations for Researchers Conducting Similar Studies

Researchers who intend to build on this work and run similar studies should consider extending AppMonitor to record attributes that would aid the analysis and resulting characterisations of the data. Five areas for improvement are identified: recording the size and position of the document window, identifying when the document has focus, anonymously recording normal key presses in Word, recording changes to Word's document length and identifying the source of document opening.

Properties of the user's screen(s), window size and position play an important role in how the content is presented. It is becoming more common for computer users to have multiple monitors, with widescreen dimensions now the norm. The rotation (portrait or landscape) of the monitor and the resolution will impact on the dimensions of the document window and in turn the navigation. These attributes, along with the the size and position of the document on-screen, should be recorded. Combining this data with logs of the users' focus will allow researchers to identify optimal settings and infer more about the tasks undertaken. Recording window focus will also allow more accurate estimates of document interaction time.

Logging more information on keyboard use and changes to a Word document's length would deliver more conclusive evidence of document editing activities taking place. While it is still undesirable to record all information regarding key-presses (for privacy reasons), identifying when keys are pressed would provide useful evidence of document editing (something that cannot currently be reliably identified). Simply recording a *key down* and *key up* event, *without* specific key information would suffice. An unobtrusive method of recording changes to the document's length would also aid this task. This data would in turn provide more concrete evidence on the reasons for the differences observed between these two applications—do they exist because of the different tasks at hand (reading vs. writing) or do they exist because of the different applications (and hence navigation tool-sets available).

Finally, information relating to the source of the document opening, be it a double-click from a Windows Explorer or an *open* command from a web-browser, would provide interaction designers with a useful background for developing more efficient systems for accessing content.

Future researchers may also want to investigate more complex data-mining procedures to identify within-document and cross-document patterns. Such analyses could identify similar patterns of navigation when the same document is re-opened or if actions are similar when tasks of same type are being performed.

5.12 Summary

This chapter has presented an empirical characterisation of electronic document navigation, over the period of 120 days, for 14 users of Microsoft Word and Adobe Reader. The primary findings of this study are that:

1. Three tools are used for the majority of vertical navigation: the mousewheel, the scrollbar thumb and the paging keys.
2. Tools that are specifically designed to increase user efficiency, such as bookmarks and search tools, are rarely used.
3. Half of users' navigation actions can be generalised across applications; the remaining half modify their tool selection based on the application.

4. Within-document revisitation is a commonly performed action, but tools that support this tasks are rarely used.

This characterisation lays the foundation for further research in this area. Other researchers can learn from the experiences reported here and build on these results with more widespread studies of within-document navigation. Many such studies should enable researchers to accurately model this complex area of interaction.

This chapter presented a large number of empirical data points. One shortfall of this characterisation is the lack of information describing the reasons users chose particular tools for particular tasks. The following chapter addresses this issue in two task-centric observations of electronic document navigation.

Chapter 6

Why do Users Navigate in a Particular Manner? Two Task-Centric Observations of Electronic Document Navigation

USERS make intentional choices when employing document navigation tools. To complement the lower-level empirical characterisation presented in the previous chapter, two task-based observations of the use of electronic document navigation tools were performed. These studies aimed to understand the reasons behind navigation tool selection and to explore users' knowledge of advanced navigation techniques. This will provide explanations for the tool use observed in the previous chapter. An understanding of the users' thought processes provides designers with insights into why particular tools are over-used and why some remain unused, and provides a platform for the development of new or improved navigation mechanisms.

To understand tool choices in particular situations, the first study asked participants to complete a series of specific navigation tasks, in Microsoft Word and Adobe Reader, while AppMonitor recorded their actions. To provide feedback on *why* tools were selected, the second study required participants to complete a series of tasks, while providing think-aloud feedback on the reasons for employing particular navigation tools.

The results and discussion of these two studies are presented in parallel. Many of the tasks used in the two studies were intentionally similar, allowing the subjective opinions from the interviews to complement the empirical data collected by AppMonitor in the first study. For ease of expression, these two experiments will be referred to as the *task study*, and the *interactive think-aloud interview session* or *interactive session* for brevity.

The tools employed for achieving the goals in the task-study closely matched

those applied in the interactive sessions, providing an across participant-set validation of the collected results. Further, several of the observations in the longitudinal study, such as the lack of use of advanced navigation tools, were also observed in these two studies.

Briefly, these studies found that the majority of participants did not use or were not aware of many of the advanced navigation techniques available, with a small subset of tools applied in most situations. Interviewee feedback indicated tools were most often selected as they were the most efficient method known to the individual for completing the task. Participants were often surprised when more efficient tools were demonstrated. Increased use of these tools could be encouraged by interfaces helping users to transition from beginner or intermediate users to experts.

The structure of this chapter is as follows. First, the goals of both studies are outlined. Second, details of the task study are introduced, including the apparatus, experimental design and participants involved. Third, the interactive session procedure is described. Fourth, the task-set, many of which overlap between the two experiments, is detailed. Fifth, the results from both studies are presented, in parallel. Sixth, subjective feedback from the participants in the task study and freeform comments from the interactive session are distilled. Finally, the chapter concludes with a summary of the presented results and areas for future work.

6.1 Study Goals

These studies aim to understand the reasons behind navigation tool selection and to explore users' knowledge of advanced navigation techniques. This will provide explanations for the tool use observed in the previous chapter.

The task study uses a constrained set of activities to observe the navigation tools employed. Tasks are selected to provide an 'ideal use' scenario for advanced navigation techniques. Use of generic navigation tools in these situations is indicative of users' ignorance of specialised tools.

The interactive sessions aim to complement the task-study by understanding the specific reasons particular tools were employed. In a similar manner to the task study, users were asked to complete a particular task. Immediately following this task, they were asked to provide feedback on their reasons for their tool selection.

6.2 Task Study

The two studies were conducted under different conditions using two mutually exclusive participant sets. This section describes the apparatus, experimental design and procedure, and the participants for the task study. The interactive session procedure is described in Section 6.3.

6.2.1 Apparatus

The full-screen experimental interface (Figure 6.1) consisted of two parts: the task cueing region (300×800 px) and a document navigation system (980×1024 px).

To begin a task, participants pressed the *Next Task* button. This displayed the instruction text for the next task and opened the required document in the correct interface. At this stage, the document navigation system was hidden by a blank pane which contained a copy of the task instructions and a *Start Task* button. Participants clicked this button to expose the interface and begin the task. To end a task, participants pressed the *End Task* button. This closed the document navigation system and enabled the *Answer* text box. An answer to the question could then be typed, before moving to the next task. Participants were not permitted to return to the document after the *End Task* button was clicked. No feedback was given as to the correctness of the provided answer.

Task times were measured from when the *Start Task* button was pressed and ended when the *End Task* button was pressed. The length of time required to type the answer was ignored.

The two navigation interfaces used were Microsoft Word 2003 and Adobe Reader 7. AppMonitor logged participants' actions while they completed the required tasks. Full logging was enabled, recording all user input via the keyboard and all mouse movement, in addition to the actions recorded during the longitudinal study. The data from the two interfaces is not merged during some of the analyses to allow comparisons between the tool sets that each interface offers.

The experiment ran on a desktop computer with a 2.8 GHz Pentium 4 processor running Microsoft Windows XP. Input was received through a Microsoft optical wheel mouse.

The experimental system controlled all of the task cueing, opened the applications and documents when required, recorded the answers provided by partic-

ipants and logged task completion times. The log files from AppMonitor were linked to answers and task times during post-experiment data analysis.

6.2.2 Experimental Design and Procedure

The sessions began with an explanation of the purpose of the study, the experimental setup using unmodified versions of Microsoft Word and Adobe Reader, the data logging and the tasks participants would be asked to perform. Participants read and signed a consent form (Appendix E.1) and were provided instructions for completing the tasks on-screen as well as by verbal explanation (Appendix E.2). Half of the participants completed all tasks with Microsoft Word first, the remainder completed their tasks with Adobe Reader first. Beginning with their assigned application, each participant completed one un-timed practice task to ensure they were familiar with the interface controls and how they were to provide answers. All tasks (see Section 6.4) were automatically cued by the system. At the conclusion of the experiment, participants completed a demographic information form and questionnaire (Appendix E.3).

Each experimental task was performed twice in each interface; once in a short document and once in a long document. The short documents were technical articles, 16 pages in length and the long documents were manuals that were 70 pages in length. All tasks in the short document were completed in the first interface, followed by the tasks in the long document in the same interface. After completing all of the tasks with one interface, the tasks were then completed with the second.

6.2.3 Participants

The 37 volunteer participants were all undergraduate computer science students (3 female) who had a mean age of 22.6 years (s.d. 5.5 years). The participants' demographic information is summarised in Table 6.1. Each participant's involvement lasted approximately 45 minutes.

Demographic	Response summary
Age	22.6 years (s.d. 5.5 years)
Dominant hand (right/left)	32/5
Mouse manipulation hand (right/left)	37/0
Mean approx. hrs/week spent using a computer [†]	35.6 hrs (s.d. 12.3 hrs)
Approximate percent of hours spent working	55.3% (s.d. 25.0%)
Indication of use of Microsoft Word:	
Never	1
Very occasionally (at the most, once a month)	8
Occasionally (at the most, once a week)	17
Regularly (at the most, once a day)	9
Frequently (multiple times a day)	2
Indication of tasks in Microsoft Word (multiple selections allowed):	
To read documents others have created	24
To edit or review documents others have created	19
To create my own documents	35
Indication of use of Adobe Reader:	
Never	3
Very occasionally (at the most, once a month)	2
Occasionally (at the most, once a week)	8
Regularly (at the most, once a day)	14
Frequently (multiple times a day)	10

[†]Participants indicated a range—the middle of the range was used as input to this calculation.

Table 6.1: Demographic information for participants in the task study

6.3 *Interactive Sessions*

The interactive sessions consisted of three main phases. First, participants were welcomed, given an overview of the interactive session process and they were asked to sign a consent form (Appendix F.1). Video recording then began. Second, the participant completed a practise task followed by the evaluation tasks in the two interfaces—Microsoft Word 2003 and Adobe Reader 7. Third, after the formal tasks, participants were given the opportunity for freeform discussion about document navigation and document navigation systems, and were encouraged to air their frustrations, as well as describe other navigation systems they regularly use and the features in those that they value.

The interactive sessions used a modified think-aloud protocol to understand the reasons for navigation tool selection. The process for each task was as follows: first, the participant was asked the task question; the experimenter simply observed the participant complete the task; when finished, the participant verbally communicated their answer to the experimenter; finally, the experimenter reviewed the user's actions with them and asked the participant to provide reasons for each of those actions. This process allowed the participants to complete their tasks uninterrupted and still provide the experimenter with valuable feedback on their tool selections.

6.3.1 *Participants*

The eight interactive session participants were a subset of those who took part in the longitudinal study. This choice was intentional, allowing for better correlation between the observations recorded in this experiment and the empirical data reported in Chapter 5. The study demographic consisted of computer science staff and students—seven male and one female.

6.3.2 *Apparatus*

The interactive sessions were performed in the offices of those who had a private workspace (four participants). Those who had a shared workspace were interviewed in a separate room with a recreation of their desktop (the remaining four participants). The participants who performed the tasks in the mock-up were

Task	Study use	
	Task-study	Interactive session
Awareness of the document's length	✓	✓
Understanding a document's structure	✓	
Locating a section	✓	✓
Locating a page	✓	✓
Phrase search	✓	✓
Finding an answer	✓	
Revisitation	✓	
Region comparison	✓	✓
Visual search		✓
Zoom	✓	✓
Extremity shortcuts	✓	✓
Reading		✓
Document editing		✓
Document opening methods		✓

Table 6.2: Tasks in the task-study and interactive sessions

asked to ensure the interfaces were set-up identically to what they used on their desktop. Participants were supplied with the documents for use on a memory stick.

The interactive session experimenter followed a series of scripted tasks (Appendix F.2) complimented with a small set of resources for visual aid (Appendix F.3).

6.4 Tasks

A series of tasks was chosen to represent some of the everyday activities that are performed in document navigation systems and that were expected to expose user knowledge of advanced navigation tools. The tasks probe users' knowledge of tools from all of the task areas reviewed in Chapter 2. These include tools employed for reading, comparison, visual search and editing tasks. The tasks are listed in Table 6.2 (the full interactive session instruction script is available in Appendix F.2). Individual tasks are described immediately before the results for that activity.

To allow for fair comparison between the two document lengths, tasks that had specific within-document targets had these targets specifically chosen to be the same percentage (to within a page) of the way through the documents. For example, in the *move to page* task, the target was always 30% of the distance into the document.

6.5 Results and Discussion

The task-study provides data on users' tool choices in particular situations. The interactive sessions complement this data by exposing reasons for these tool selections. For brevity and to provide a more complete picture of user interaction for each task, the analysis and results from these two studies are presented together.

In the task-study, a total of 1628 tasks were attempted, 154 of these were discarded as they were answered incorrectly. These wrong answers were usually caused by the participant navigating to the incorrect page. Occasionally participants also indicated that they 'forgot' the answer, as the response box was only enabled *after* pressing the *end task* button. A further 34 tasks were removed as they were outside the task mean + 3 s.d. for the particular task (due to the participant becoming distracted or totally lost in the document). All interviewees completed all questions. A total of 72 tasks required responses; only nine were answered incorrectly (usually due to the incorrect counting of items).

The remainder of this section discusses the tools employed in each of the tasks and reasons provided by the participants for their use.

6.5.1 Interface Setup

The task-study was run in an undergraduate computer laboratory, with each participant provided with a default installation of Microsoft Word and Adobe Reader. At the beginning of the study, participants were invited to customise each application's interface to include or remove any toolbars, panels or task bars, so as to recreate the interface they normally used. These changes were not explicitly recorded.

Recall that half of the interactive session participants used their own workstation and half used a mock-up of their desktop. Those who used the recreation were asked to modify the provided interfaces from their default state to one that

Participant	Window maximised	Toolbars			Panels		Notes
		Standard	Reviewing	Customised	Style	Thumbnails	
1	✓	✓	✓		✓		Microsoft Word 2007
2	✓			✓		✓	
3	✓	✓	✓				
4	✓	✓					
5		✓	✓				
6	✓	✓					
7	✓	✓	✓				
8		✓	✓				

Table 6.3: Summary of interviewees’ interface setups for Microsoft Word

was identical to their own computer. The experimenter suggested that participants consider toolbars, panels and task bars, and the application size and position on screen.

The interface setups for Microsoft Word are summarised in Table 6.3. Six participants used Word in its maximised state. During the interactive sessions, participants were made aware that the experimenter would only require them to complete one task at a time. This information may have partially influenced some participants to decide to maximise the window (although none explicitly stated this). However, one participant explained that they often maximised their windows to ease navigation: “... I usually maximise it ... because I use the scrollbar so much ... say you’re web-browsing ... and you want to scroll down a whole lot, you don’t really even have to look at the side of the screen ...”. In this case, a window in its maximised state reduced the demands of acquiring the scrollbar, by making one edge infinitely big (once the cursor reaches the side of the screen, it stops).

In general, participants used Word with only a few toolbar or task pane adjustments from the default setup. The reviewing toolbar was the most common addition, giving an indication of the type of work participants often conduct—

reviewing documents with multiple authors. During the course of the interactive session, several participants temporarily added additional task panes, but these were immediately removed when the task was complete.

The same six participants that maximised Word's window also maximised Reader's window. Apart from this change, all participants used Reader in its default interface setup.

6.5.2 Awareness of the Document's Length

This task aimed to test the participants' awareness of the length of the document at hand—an attribute that can influence the choice of navigation tools. It asked: *how many pages are there in this document?*

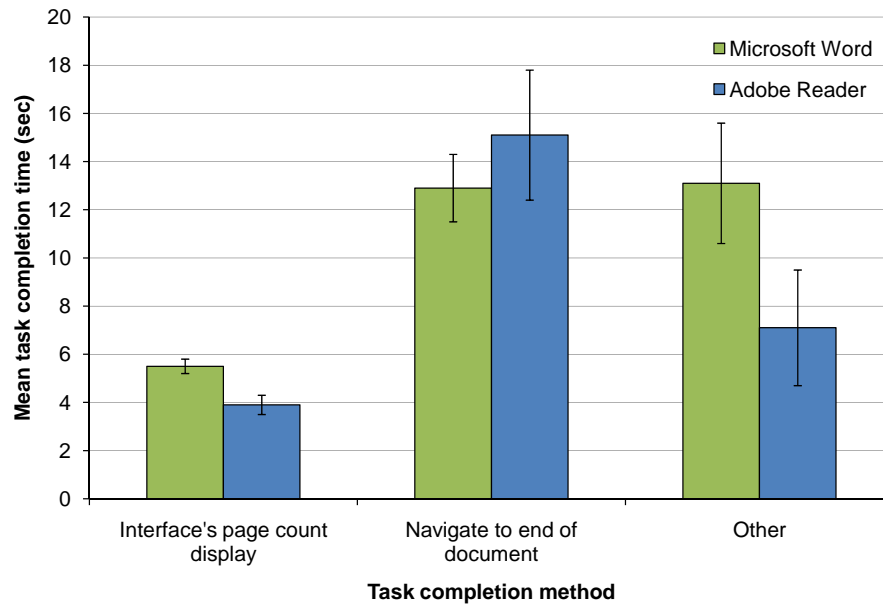
Participants used two main methods to determine the length of a document: the interface's page count display and navigating to the end of the document. The mean task completion times, by method, are shown in Figure 6.2a and the distribution of methods between participants are shown in Figure 6.2b.

In the task study, 52% of Word and 71% of Reader tasks were completed using the interface display, taking a mean time of 5.5 sec (s.d. 1.7 sec) and 3.9 sec (s.d. 2.4 sec) respectively. The majority of the remaining tasks were completed by moving to the end of the document: 37% of Word users and 16% of Reader users took this action, taking mean times of 12.9 sec (s.d. 7.1 sec) and 15.1 sec (s.d. 8.5 sec). Tasks not accounted for in these two categories were completed by using other tools, such as the *Word Count* functionality.

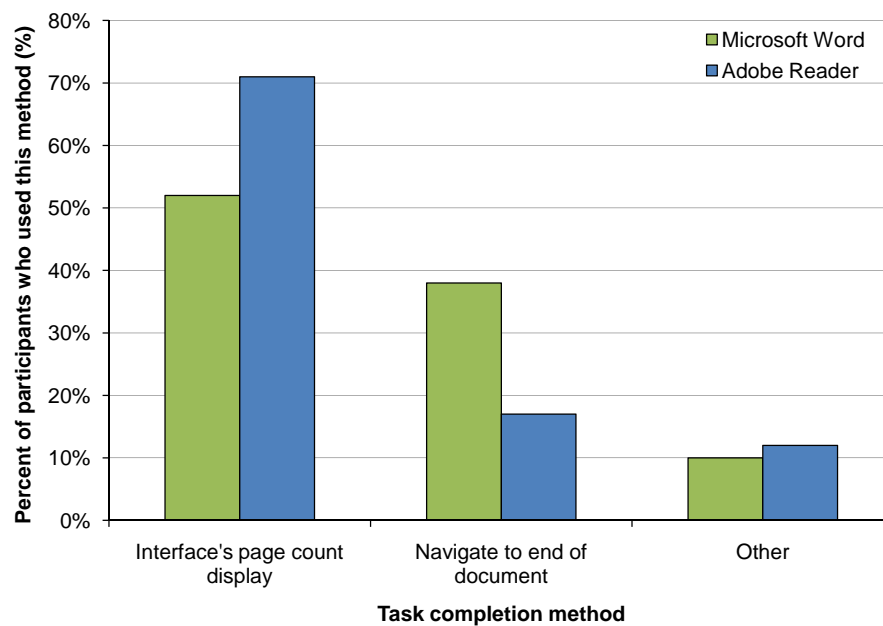
The increased awareness of interface information in Reader likely reflects the more prominent position of its page count indicator—bottom center of the screen, compared with bottom left in Microsoft Word.

In the interactive sessions, all participants correctly identified the lengths of the documents in both applications. One Word participant attempted to use the thumbnail pane (that displays page numbers under each thumbnail), but ended up using the page count indicator when they could not read the page numbers under the thumbnails. The remainder of the tasks were completed using the page count indicator on the status bar. All participants utilised Reader's page count, set in the bottom center of the interface.

This task demonstrated that 50–70% of participants are aware of page count



(a) Mean time for determining the length of a document. Error bars show standard error.



(b) Distribution of methods for determining the length of a document

Figure 6.2: Document length task result summary

indicators showing the exact document length. However, the number of pages is not the only method for understanding the size of the document. Participants may also use implicit cues, such as the extent of the scrollbar thumb, to provide a rough estimation of the document length. Tasks described later in this section will examine tool selection differences that arise in different length documents.

6.5.3 *Understanding a Document's Structure*

The goal of this task was to determine whether participants were aware of tools that display the document's structure. This information gives the reader an overview of the content and can aid the user when locating regions of interest. Both applications used in these studies include tools to overview this structure—Word provides a *Document Map* and *Outline View*, and Reader provides the *Bookmarks* functionality. The documents themselves may also provide a *table of contents* for this purpose. In these studies, all of the documents had tables of contents which were *not* removed for the tasks. The questions for this task were of the form: *how many sections (not sub-sections) are there in this document?*

In the task study, none of the Word tasks were completed with the aid of document structure tools. Participants instead used the table of contents and/or tools such as the scrollbar thumb and mousewheel to navigate through the document to understand its structure. In Reader, the *Bookmarks* tab, which displays the structure of the document, was used in 17% of tasks. All other tasks were achieved by using the contents page and/or moving through the document.

The interactive sessions probed user familiarity with advanced navigation tools. This questioning provided an explanation for the lack of use of the tools specifically designed to aid this task. Only one of the eight participants was aware that Word's document map existed, but indicated they never used it. Most participants were familiar with Reader's bookmark tab; however, as was seen in the longitudinal study, this is far more likely to be employed if it was visible as the document was opened: "if it's there I might use it ... but I hardly ever open it". This tab was not automatically opened in this task. Having these document structure tools visible, especially in larger documents, would increase the likelihood of users employing these advanced navigation techniques.

6.5.4 Locating a Named Section

For this task, participants were required to move to a named section in the document. The aim was to determine whether participants employed document structure tools (as described was expected in the previous task) or whether find and search functionality was preferred. These tasks were cued by asking questions such as: *how many bullet points are there in the “Performance Metrics” subsection?*

In the task-study, the *Find* functionality was used to locate the correct section in 34% of tasks in the short Word document (mean time 30.5 sec, s.d. 14.0 sec) and 94% of tasks in the long Word document (mean time 54.1 sec, s.d. 40.5 sec). In Reader, the find functionality was also applied in 31% of the short document tasks (mean time 36.8 sec, s.d. 14.3 sec) and 94% of the long document tasks (mean time 57.1 sec, s.d. 44.1 sec). None of the Word tasks were completed using document structure tools and only 11% of Reader tasks were attempted using the bookmarks functionality, taking a longer mean time of 58.9 sec (s.d. 39.2 sec).

Participants showed a greater preference for the find functionality when using long documents. In the short documents, the scrollbar thumb was used for 69% of Word tasks (mean time 28.9 sec, s.d. 11.8 sec) and 53% of Reader tasks (34.8 sec, s.d. 15.5 sec). Participants may have realised that continuing to use the scrollbar thumb in the long document was inappropriate. This is a good demonstration of users switching navigation tools in documents of different lengths.

Heavy use of find functionality was also observed during the interactive sessions. In the Word document, all but one participant initially attempted to use find using the Ctrl-f shortcut. In this case, the participants were stifled by the large number of occurrences of the section name in the text. Five participants then resorted to using the table of contents and one the scrollbar thumb. The participant who did not employ Ctrl-f immediately used the scrollbar thumb. Many participants commented that their first instincts were to use a search. The table of contents became their ‘fallback’ method for completing the task.

No such problems were encountered in the Reader tasks—five participants successfully located the section using the find or search functionality. Two used the bookmarks (only one successfully) and one used the scrollbar thumb. Participants commented that because they knew the title of the section, find was the most

obvious choice, as the target would be “... obvious if I found it ...”

6.5.5 *Locating a Page*

This task aimed to determine how aware participants were of tools that aid specific page location. It was expected that users would employ the *goto* tool to quickly move to the required page. Participants were provided with a specific page number and a simple question to answer about that page. For example: *what is the name of the heading at the top of page 26?* Locating specific pages is necessary when users themselves remember a page number, or external or internal links provide a reference to a particular document location.

In the task-study, 60% of the Word tasks used the scrollbar thumb to move to the required page, taking an average of 14.6 sec. The *goto* functionality was used in 13% of tasks, taking a longer 19.3 sec. Paging and other tools, such as rate-based scrolling, accounted for the remaining tasks. More in line with expectations, *goto* tools were most commonly used in Adobe Reader, accounting for 43% of actions, taking a mean time of 23.2 sec. Following this was the scrollbar thumb (24% of actions, 17.7 sec), paging (24%, 15.4 sec) and the thumbnails (9.5%, 15.1 sec).

Surprisingly, the *goto* functionality was the slowest method for completing this task in both applications. Closer analysis of the logs revealed that several *goto* users spent many seconds searching through the menus to find this functionality. One participant spent 16 seconds searching through Word’s menus, selecting the following top-level items before the *goto* entry: Tools → Format → Insert → Format → Insert → View → Edit → File → Edit → Go.To... Searching in this manner indicates that the user was aware this function existed; however, the long search time indicates this user does not regularly employ the *goto* function.

During the interactive sessions, a very similar pattern of use emerged—Word tasks were predominately completed using the scrollbar thumb and Reader tasks through the editable page counter at the bottom of the interface. Several Word participants cited the small page counter that appears beside the scrollbar when dragging the thumb as their reason for using this tool—they could quickly and accurately determine when they had reached the correct page.

Five of the Reader participants typed the required page number into the ed-

itable page counter on the interface. The primary reason conveyed for this behaviour was that it took them “straight to the page”. The remaining participants used the scrollbar thumb (2) and paging buttons (1), because they were unfamiliar with methods for moving immediately to a numbered page.

When locating a page in the document, users of the scrollbar thumb outperformed those who employed specialist *goto* tools. Users of the scrollbar demonstrated their familiarity and accuracy with this tool; goto users spent their time hunting for the correct menu item, re-orientating themselves after the jump action and switching to another tool for fine scale adjustment to obtain the required answer.

6.5.6 *Phrase Search*

The goal of this task was to determine whether participants were familiar with the *find* and *search* tools. This feature is quoted as one of the main advantages of electronic media over paper [32], despite its apparent lack of real-world use [137] (the longitudinal study also observed little use of this feature). Participants were required to find the definition of an abbreviation. For example: *what does the abbreviation SAGE stand for?* Interactive session participants were also asked to count the number of occurrences of a particular phrase.

The majority of phrase search tasks were completed using the find and search tools available in these applications. This confirms that users are aware of these tools and understand how they should be applied.

In the task study, Microsoft Word users employed two search techniques. First, the Ctrl-f keyboard shortcut was applied in 73% of tasks, with a mean completion time of 16.2 sec (s.d. 8.3 sec). Second, the Edit → Find . . . menu item accounted for 25% of tasks, with an average completion time of 28.2 sec (s.d. 10.3 sec). One person used the scrollbar thumb to locate the phrase.

Adobe Reader users applied a more diverse range of methods for locating the phrase. The Ctrl-f keyboard shortcut was the most popular method used for completion—51% of tasks applied this technique, taking an average of 13.7 sec (s.d. 5.0 sec). Following this was the use of an already visible find dialog box (27.5% of users taking 11.4 sec), the find menu item (13% of users 31.4 sec) and users of search (5.8% taking 29.7 sec). Two users employed other search methods.

The interactive sessions contained two tasks that were explicitly designed to test the participants' knowledge of phrase search functionality—the first required the user to simply locate a phrase. The second required users to count the number of occurrences of a particular phrase in the document. This task may initially sound artificial, however searching in this manner can be useful when users wish to identify the relevance of the document to a particular topic.

In the interactive sessions, as expected, all participants used find functionality in the task to locate a particular phrase, in both applications.

When counting the occurrences of a particular phrase, two Word participants used find functionality to step through the matches. Four participants used the *replace* functionality, knowing that it would report the number of matches. Two employed the *Highlight All* feature, allowing them to quickly count the occurrences. In Reader, four participants used the *search* functionality that returned a list of matches and counted those, the remaining four used the find functionality to step through the phrase matches.

In all of these phrase search tasks, participants employed the find and search functionality, confirming user knowledge of the operation of this function. However, the lack of real-world use identified in previous research, and during the longitudinal study, confirms that users struggle to deploy this tool in their work. Loizides and Buchanan [137] noted one reason for this observation was users' lack of knowledge of their search target. A further conjecture is that users may not confidently know *when* they should employ the find tool. The observational nature of these phrase search tasks potentially triggered participants to applying these tools. In their normal work, users may instinctively reach for their default navigation tool, relegating find tools to their 'last resort' search mechanisms.

6.5.7 *Finding an Answer*

Large reference documents, such as instruction manuals, are used to locate the answer to a question or to find a solution to a problem. When performing such a task, the user may or may not know an exact phrasing that will allow them to efficiently use within-document search tools to locate an appropriate answer. The aim of this task was to investigate how users coped when searches returned many results. The questions contained sufficient keywords to allow task completion,

but find utilities were hampered by many search results. For example: *how many levels of rigor does the document define in relation to formal methods?* Searching for the phrase “levels of rigor” rendered no results, “rigor” matched 15 times and “levels” matched 25 times.

In short documents, approximately half of participants *did not* employ any find tool; in long documents, virtually all participants used search functions. Not using the search functionality was not significantly detrimental to users’ performance: tasks that used find had a mean time of 94.6 sec and those that did not had a mean time of 97.7 sec. This small completion time difference is explained by the large number of matches users searched through. Find actions had a mean of 10.1 (s.d. 17.2) search commands (for example, *find next*) per document.

This result highlights one of the inflexibilities of current find facilities in document navigation systems—search features are often exact match only (note that Microsoft Word does provide wildcard matching, but it was not employed by any of the participants). This result confirms Loizides and Buchanan’s [137] findings that regular expression matching, single word matching, spelling assistance and stemming would significantly improve the efficiency of find actions.

6.5.8 Revisitation

The goal of the revisitation tasks was to understand how navigation tool use changed as participants became familiar with the document and with that, how task completion time decreases as a document position is learned. Participants repeatedly (but possibly unknowingly) searched for the same target three non-consecutive times. A thumbnail image of a page was shown in the task-cuing interface. Each iteration provided the user with a simple question about the page, for example, *what is the second word in the sentence immediately after the figure on this page?* A different question was associated with each iteration to ensure participants had to move to the page to provide the correct answer.

Three results are presented for the revisitation tasks: task completion times, the tools employed for task completion and application differences.

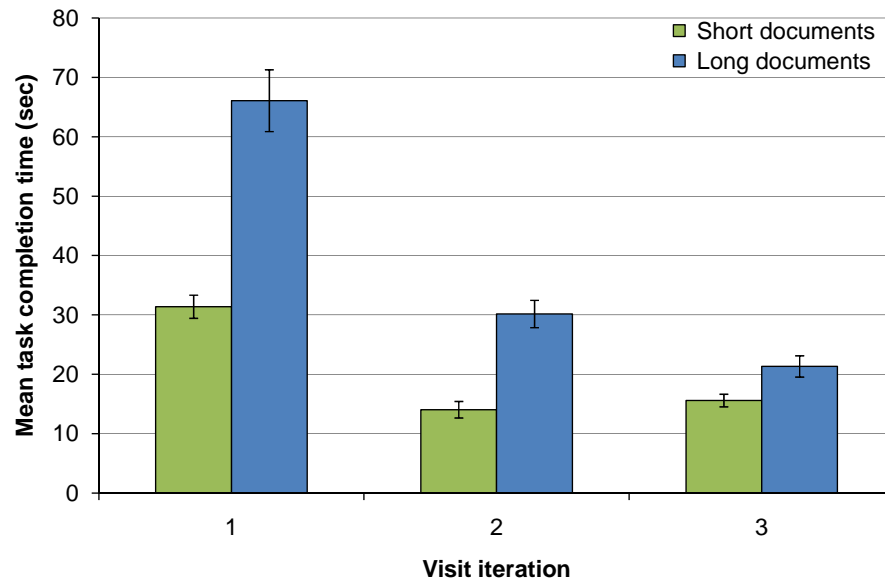


Figure 6.3: Revisitation task completion times. Error bars show standard error.

Task Completion Time

The task completion times for these tasks were analysed using a $2 \times 2 \times 3$ ANOVA, with factors interface (Word or Reader), length (short or long document) and iteration (1, 2 or 3). A summary of the task completion times is provided in Figure 6.3.

Tasks in short documents were completed significantly faster, taking a mean time of 20.9 sec (s.d. 13.4 sec) compared with long documents at 38.7 sec (s.d. 31.6 sec), $F_{1,36} = 97.0$, $p < 0.001$. This reflected the greater number of pages that participants scanned before the target was reached (see the discussion of navigation tools used in this task). There was also a significant difference for the factor iteration, with iteration one taking 48.5 sec (s.d. 32.6 sec), iteration two taking 22.3 sec (s.d. 16.4 sec) and iteration three taking 18.6 sec (s.d. 11.8 sec), $F_{2,72} = 118.9$, $p < 0.001$. As expected, users were learning the position of the target in the document over the three iterations.

Tools Employed for Task Completion

A summary of the primary navigation tools used to complete these tasks is shown in Figure 6.4. In this figure, the bubble widths are indicative of the percent of

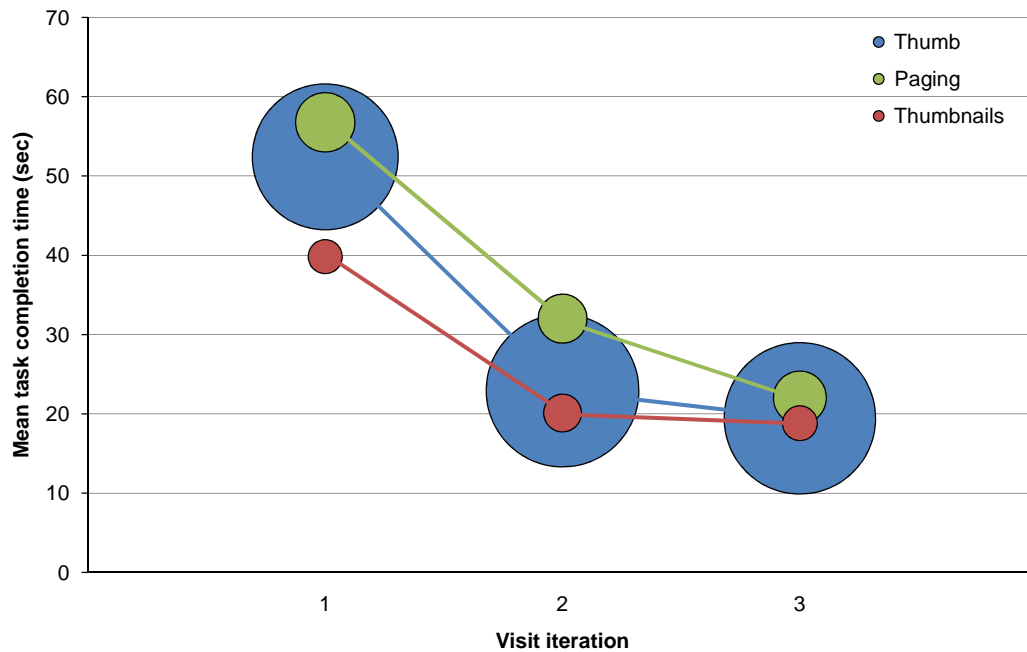


Figure 6.4: Tools used for revisitation tasks. Bubble widths are indicative of the percent of tasks completed with the tool.

participants who used this tool. Thumbnails were the fastest tool for completing the task, but were only used by a small percentage of people. The scrollbar thumb was the second fastest and was used by the greatest number of people. The tools employed by participants remained nearly constant through the three iterations. This indicates participants did not try to improve their search time using different tools, but were able to perform the same task significantly faster, with the same tool, as they became more familiar with the target's position.

Application Differences

Tasks in Reader were completed significantly faster than in Word, with a mean time of 25.6 sec (s.d. 17.1 sec), with Word tasks taking 34.1 sec (s.d. 31.7 sec), $F_{1,36} = 18.3$, $p < 0.001$. This is explained by the lesser use of the scrollbar thumb and increased use of efficient tools (the thumbnails) in Reader. This tool choice is likely due to users having a wider knowledge of Reader's tools than Word's—a later question in the interactive sessions (see Section 6.5.16) found that more

people were aware of the thumbnails in Reader than they were in Word.

6.5.9 Region Comparison

Region comparison tasks required participants to identify the difference between two non-consecutive paragraphs in a document. The goal of these tasks was to probe user awareness of advanced navigation tools that can efficiently compare two document locations. This situation can arise when writing or proof-reading a document—the reader may need to refer between the position of writing and material earlier in the document to avoid repetition or to form a coherent argument. It was expected that Word users would employ the split-view tool, while Reader users would take advantage of the view navigation tools to quickly move between document positions.

Task-study

The advanced navigation tools were only used in four of the 105 correctly answered region comparison tasks, despite their significantly shorter task completion times. All of this use was with Word's split-view. As shown in Figure 6.5, the users of the split-view functionality completed the task significantly faster than all other techniques, taking a mean time of 65 sec (s.d. 13.1 sec). This was 46 sec quicker than the second fastest method, paging between the document positions. Except paste (see below) the remaining techniques required the user to move backward and forward between the two positions, explaining the greater task completion times.

The figure shows copying and pasting the content, so it was consecutive in the document, as the most popular method for task completion (even some users of Adobe Reader, in which you cannot modify the content, copied the two paragraphs into a Word document); however, it was the third slowest method for achieving the desired result. Further analysis of the log files revealed that this tactic was often employed as a 'last resort' when users became frustrated with moving between the two paragraphs.

All tool categories had mean task completion times greater than a minute. The mean times of many categories were twice that of the tasks completed with the split-view tool. It is safe to assume users who were aware of the split-view func-

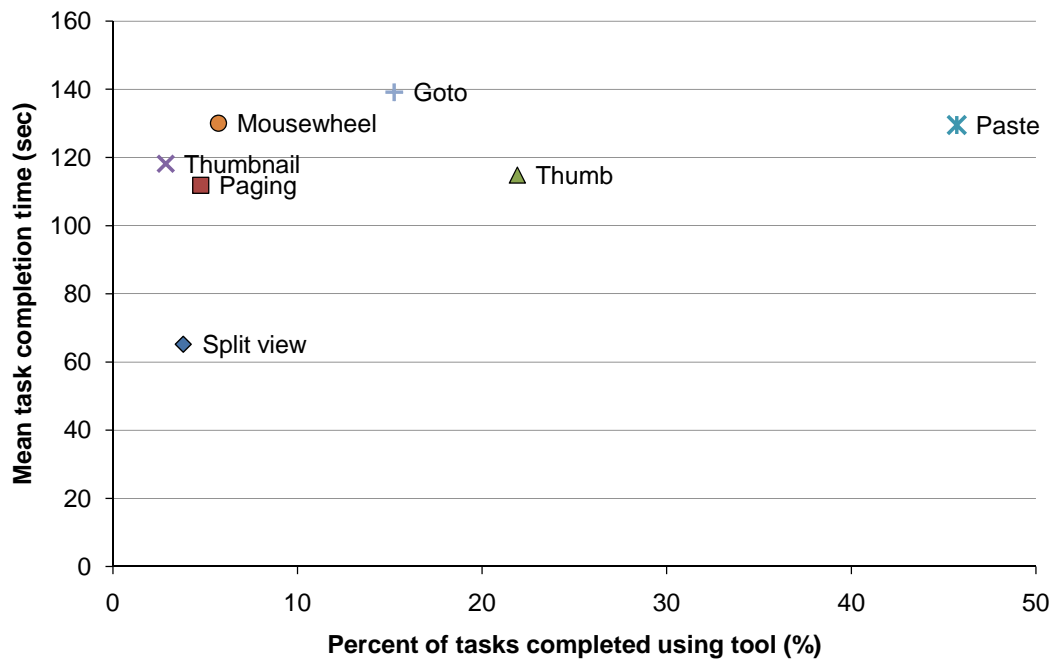


Figure 6.5: Tools and task completion times for comparison tasks. Error bars omitted for clarity.

tion would employ it in this situation, saving themselves minutes in task completion time.

Interactive Sessions

A greater percentage of interactive sessions participants used Word's split-view functionality (50%); none of the participants used Reader's previous/next view features. Three of the Word participants and one of the Reader participants who were not aware of these efficient navigation tools used satisficing strategies to complete the tasks, by copying and pasting the paragraphs into consecutive positions in the document.

Further questioning of participants revealed that only one of the Word participants who did not use the split-view was aware it existed; only two of the Reader participants were aware of the previous/next view functionality and neither considered applying it to this task.

6.5.10 Visual Search Techniques

Visual search tasks require users to identify regions of a document using only their perceptory sense. Visual search is employed in a variety of situations, including when the user does not know their exact search target, when they do not know a more efficient search strategy or, as was observed in previous tasks, to understand the document's structure. The goal of this task was to identify the techniques employed to complete visual search tasks and whether these techniques change according to the type of object the user wishes to locate.

Three of the interactive session questions asked participants to locate objects within the document. The first asked participants to find an image, the second a blank page and the third to count items that were specifically marked (for example, surrounded by a red box).

When using Microsoft Word, many participants used the same tools to complete all of the visual search style questions. These tools are summarised in Table 6.4. Note that the groups of 'b, c, f' indicate the tool was used in all three tasks. The use of tools in Adobe Reader was more chaotic (see Table 6.5). Four participants consistently used at least one tool for all visual search tasks, the remaining four varied their tool use.

In both applications, all but one participant used zooming to help complete these tasks. Targets were correctly identified as "rather big" and acquisition could occur "even if you have a very small zoom". Several participants took advantage of this by viewing multiple pages at a time. Various tools then assisted with navigation through the multi-page views. The tactics employed in this task demonstrated that users are familiar and competent at visual search tasks—possibly a by-product of users' favouring this technique over more specialised search tools.

6.5.11 Zoom

The goal of this task was to determine how familiar participants were with the zoom tools provided in the interfaces. Zoom tasks required users to manipulate the document's magnification in order to read a word displayed in a very small font. For example: *The map on the first page of the document contains a red box. What is the word that is inside the box?* As with vertical navigation, both interfaces support several methods for achieving this task.

Participant	Tool					
	Zoom out	Mousewheel	Thumbnails	Thumb	Arrow	Paging
1	b, c	f, c				
2			b, c, f			
3	b, c, f	f				
4	b, c, f	c				
5	b, c, f					
6	c	f, b		b	c	
7						b, c, f
8				b, c, f		

b = locate blank page c = count objects
 f = locate figure

Table 6.4: Tools used to complete visual search tasks in Microsoft Word

Participant	Tool					
	Zoom out	Mousewheel	Thumbnails	Thumb	Two-page view	Paging
1	b, c, f	f		c		b
2					b, c, f	b, c, f
3	c, f	c, f		b, f		
4	b, f	b	c			f
5	b, f			b, c, f		
6	c	f	b	f		c
7	c, f					b, c, f
8				b, c, f		

b = locate blank page c = count objects
 f = locate figure

Table 6.5: Tools used to complete visual search tasks in Adobe Reader

The zoom tasks demonstrated the efficiencies that expert users gain when employing advanced navigation tools. This was most clearly demonstrated in the Word task, where the Ctrl-Mousewheel technique had a mean time 7.3 sec faster than the second most efficient zoom tool, the zoom drop-down box. Zoom changes made from menu selections were the slowest. Users of non-expert tools were unaware of “straightforward alternative[s]” to visually dominant tools, such as the zoom drop-down box.

A similar pattern of use was observed in the interactive sessions. In Word, seven participants used the zoom drop-down box and one used the Ctrl-Mousewheel technique. The drop-down box was used as participants were unaware of a “straightforward alternative”. However, the zoom results were not completely satisfactory, with users noting the lack of panning: “I wanted to drag it there, but Word doesn’t let me”. In Reader, four participants used the zoom in push button, three the magnifier tool and one the Ctrl-Mousewheel technique. The push button was applied most often for its ease of use.

Finally, when asked to return the document is a normal reading zoom, identical tool use as reported in the previous paragraph was observed—in Word, seven participants used the drop-down zoom adjustment box and one the Ctrl-Mousewheel technique; in Reader, five used the zoom-out push button and one each used the drop-down menu, Ctrl-Mousewheel and the *page height* button.

6.5.12 *Extremity Shortcuts*

Extremity shortcuts allow the user to quickly move to the start or end of their document. On its own this action rarely completes a task, however it can increase the efficiency of tasks such as navigating to the table of contents or references sections at the beginning or end of a document. The goal of this task was to determine whether users were aware of the shortcuts that support long distance navigation actions. Participants were asked a question regarding the last page in the document. For example: *How many “Related Links” are displayed on the final page of the document?*

In the task-study, this task was generally completed in two distinct phases: an

open loop phase¹ (to move to the end of the document) and a *closed loop*² phase, to accurately position the document to answer the question. Only the open-loop portion of this task considered.

The scrollbar thumb was the most commonly employed method for moving to the end of the document. In Word, this accounted for 77% of the use (mean use time 2.1 sec, s.d. 1.1 sec). In Reader, 40% of users utilised the scrollbar thumb (3.7 sec, s.d. 5.6 sec) and 39% of users took advantage of the *Last Page* push button (0.4 sec, s.d. 0.4 sec). The fastest open loop method recorded for moving to the end of the document was the keyboard shortcut Ctrl-End (or simply End in Reader), taking 0.2 sec, however only 10% of Word participants and 8% of Reader participants applied this tool.

A similar pattern of use emerged during the interactive sessions. In Word, six participants used the scrollbar thumb to move to the end, as did four of the Reader participants. The remainder used the Ctrl-End keyboard shortcut or the *End* interface button in Reader. Participants who used the thumb commented that it was “the fastest way to go a very long distance”, those who used shortcuts stated that it “seemed obvious” to use this method.

In Word, returning to the beginning of the document was achieved using exactly the same mechanisms (with Ctrl-Home instead of Ctrl-End). Scrollbar thumb users commenting that “you can slam it up to the top and it’s very easy to get there fast”. In Reader, five used the scrollbar thumb, two the Ctrl-Home shortcut and one the *First Page* interface button.

This task will likely form part of a larger activity. For this reason, task context plays an important role in the “fastest” method for moving to the start or end of the document. The most efficient mechanism is likely dependent on the preceding or following action. For this action, keyboard shortcuts have the raw fastest time. However, if the user must also release the mouse and move their hand to the keyboard (or the reverse following the action), the quickest and most comfortable method—using the scrollbar thumb—will take precedence.

¹Open loop: no feedback required to complete the task.

²Closed loop: completing the task based on feedback from the display.

6.5.13 Reading

Reading tasks require the user to move slowly through a region of the document. The goal of this task was to understand the reasons users chose particular tools to navigate through a region of a document while reading elements aloud. Questions took the form: *section 5.6 contains several subsections, indicated by bold text. Please read out the titles of each of these subsections.*

The mousewheel and scrollbar thumb were the most commonly used reading tools. Word interactive session participants were evenly split between these two tools for completing the reading task (four participants each). Participants indicated the mousewheel was good for “fine grain movement” and that they had “more sense of control” using the mousewheel than other tools. One participant who used the thumb commented that they simply continued using it after the long distance movement to the required section (and never released the button). Reader participants were more diverse in their tool choice, with three participants using the mousewheel, three the scrollbar thumb and one each the scrollbar arrows and the hand-tool. Several participants commented their tool selection was the “easiest” or least effort for completing the task.

The commonality between all of the tools employed is their continuous linear nature for moving slowly through the document—tools such as page up/page down were *not* used in this situation. More extensive observations are required to determine how other types of reading, such as receptive reading, would employ electronic document navigation tools.

6.5.14 Document Editing

Editing tasks required the user to precisely locate positions of interest and make the requested changes. This task aimed to understand the navigation techniques employed when editing was taking place. Time restrictions on the interactive session prevented observation of a user creating a full document from scratch. Instead, participants were asked to make edits that included correcting the year of a date, rephrasing passages and deleting sentences. These tasks were only performed in Microsoft Word—Adobe Reader does not support editing.

The editing changes were indicated on the document using Word’s *comment* functionality. Participants were also provided with a printed version of the docu-

ment (with the required changes marked) at the beginning of the task.

Participants employed two primary techniques for navigating between the required editing points: the built in reviewing tools for moving between comments and general navigation tools such as the thumb, paging and mousewheel. The use of the reviewing tools to complete this task was an unexpected side-effect of marking the required edits with comment balloons—it had not been the intention to test the users’ skills with reviewing tools. However, three of the eight participants used the *next comment*, *browse by comments* and the *reviewing pane*, commenting that they “took me straight to what I was concerned with” and provided a “discrete navigation path” that meant they would not miss any of the required edits. Users who applied more general navigation techniques cited their expectations of regularly encountering an edit—one user who paged through the document indicated that this method was “the fastest way to find the next one” because the balloons were “quite visible” (reducing the task to a visual search exercise).

The use of electronic editors for writing and proof-reading documents is a large area of study in itself; however, this small task has shown that experienced users do employ advanced reviewing tools when another author has indicated editing positions.

6.5.15 Document Opening Methods

In the longitudinal study, it was observed that few people opened documents from within the applications. In the task-study, documents were opened automatically, to ensure the correct documents were used and to ensure that task completion times were accurately recorded. In the interactive sessions, participants were provided with a memory stick that contained the required documents. The only instruction provided was the name of the required document and its containing sub-directory. User actions were observed over the course of the interactive sessions to understand patterns of use. When required, users were simply asked: *can you please open <document-name>?*

Participants were asked to open a total of nine documents during the interactive session. The majority of documents were opened by double clicking on the correct file in the Windows Explorer application. One participant opened four of their documents from the File → Open menu. One further participant attempted

three times to open the required document from the *Open* menu item and once from the *Open* toolbar button. They quickly gave up and returned to Windows Explorer when the open dialog did not display the correct directory to open the next file. The file system navigation required to locate particular files may explain some of the lack of use observed during the longitudinal study. Users may instead prefer to leave Windows Explorer windows open at the correct location for their current task.

6.5.16 *Familiarity with Advanced Navigation Tools*

The longitudinal study showed that a small set of navigation tools are employed the majority of the time. At the conclusion of that study, the extent of the participants' knowledge of more advanced navigation tools was unclear—were they aware of them, but never used them or did they not know they existed?

The preceding tasks have shown occasional use of advanced tools but have often failed to observe participants regularly and competently applying advanced tools in their ideal-use situations. After completing the tasks with one interface, participants were asked to demonstrate a selection of the more advanced navigation tools.

The participants—recall these were all people who participated in the longitudinal study—were first asked to demonstrate the use of the selected tool. If they were aware of the tool, they were then asked to provide an approximate rating of their frequency of use of this tool. Participants who were not familiar with the name of a tool were provided with a brief explanation of what the tool can achieve, in order to jog their memory. Competency ratings were entered on a five point Likert scale (1 = not familiar, 5 = competent); Frequency ratings used the same scale to answer the question *I frequently use this tool* (1 = disagree, 5 = agree).

The tools selected from Microsoft Word were those that were previously observed as less well-known and less used and those that had the potential to allow efficient completion of the tasks in these two studies.

The results from the competency and frequency of use ratings for Microsoft Word are shown in Figure 6.6. Users were the most confident in demonstrating the *split-view*, *rate-based scrolling* and the *Ctrl-Mousewheel* techniques, each scoring an average between 3–3.5. Recall that the Likert scale questions were rated on

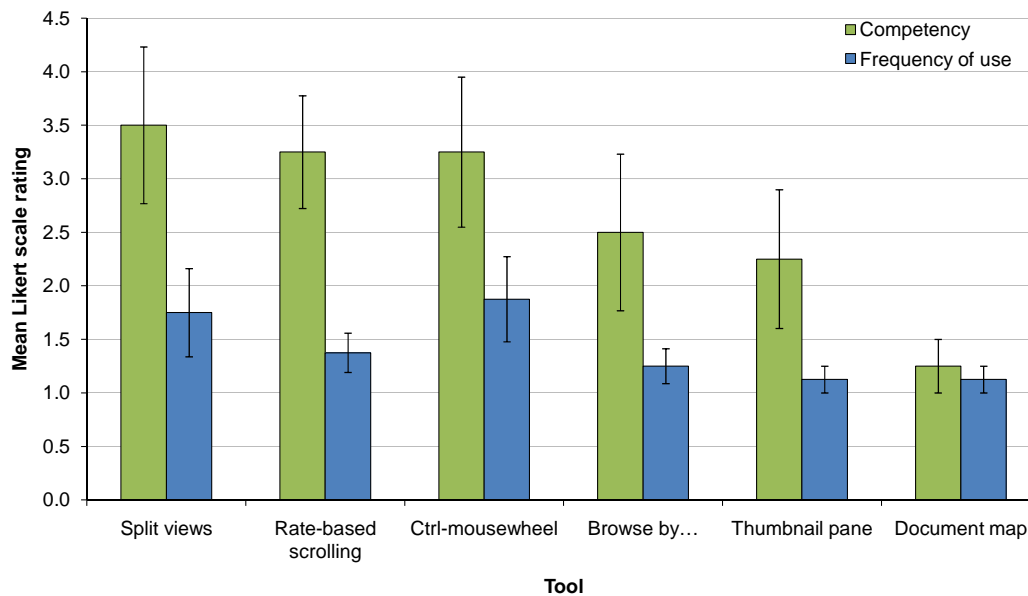


Figure 6.6: Competency and frequency of use of selected navigation tools in Microsoft Word. Error bars show standard error.

a scale out of five—these averages do *not* demonstrate high competency of the study group, but rather slightly above average. The *Browse by ...*, *thumbnail pane* and *document map* tools scored worse, with only one participant aware of the document map. Few of these advanced navigation tools were regularly used—all falling below a 2.0 frequency of use rating.

The tools selected for demonstration in Adobe Reader consisted of common and less-known tools. The *find* and *search* tools were included to investigate any differences in users' competency with these two similar tools, while the remainder were chosen for their potential utility in the task study and the interactive session.

The results of the competency demonstration and frequency of use for Adobe Reader are shown in Figure 6.7. Users were competently able to demonstrate the *find* and *search* tools, and use of the *bookmarks* and *thumbnails* tabs. The reported frequency of use for the *find* and *search* tools was the highest, with less for the bookmarks and thumbnails. Again, the participants reported frequent use of the find and search tools, a result that matches Loizides and Buchanan's [137] findings, even though the longitudinal study observed only 137 occurrences of their use across all users. The remainder of the tools could not be classed as

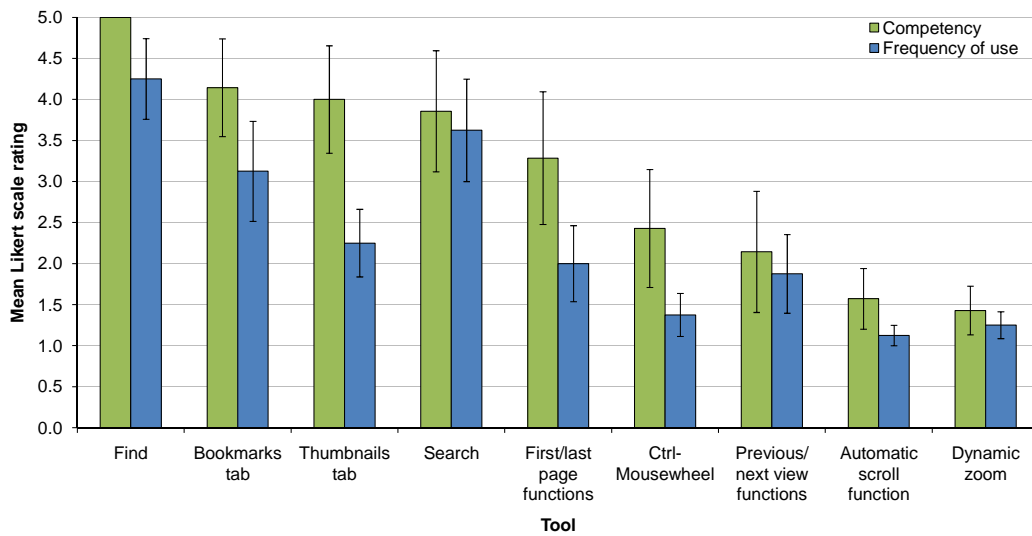


Figure 6.7: Competency and frequency of use of selected navigation tools in Adobe Reader. Error bars show standard error.

competently demonstrated or frequently used. Participants who were not aware of these tools were shown their operation.

The observation that expert users are *not* aware of many of the advanced navigation tools is consistent with the observations in the longitudinal study and task-study. This provides suitable validation of the results from these two studies—the lack of use of advanced tools was *not* a product of study time or duration.

There are several explanations for users not having an awareness of these tools. First, these studies often observed satisficing strategies for achieving tasks—if a familiar tool (such as the scrollbar) allows the user to complete the task, they have little motivation to put the effort into learning another tool. Second, the efficiencies gained by more advanced tools are potentially too small to warrant switching tools. Third, previous versions of these applications may not have contained some of these tools—long time users may never take the time to investigate new tools in new software versions. Fourth, the applications provide little aid in helping users discover these tools and advance from beginner to expert users. A much longer and larger study is required to understand which of these conjectures most heavily influences the low use of advanced tools.

6.6 Task-Study Subjective Analysis

At the conclusion of the task study, participants were encouraged to provide feedback on the tasks they had performed and to make general comments regarding document navigation. A freeform discussion of similar issues took place at the conclusion of the interactive sessions and is reported in the following section.

6.6.1 Task Difficulty

Participants were asked to rate their agreement with a statement indicating they found the tasks “easy to complete” when using each of the interfaces and document lengths. A 5-point Likert scale (1 = disagree, 5 = agree) was used to record their responses.

Participants found the tasks in the long documents significantly harder than those in the short documents. Short documents had a mean ease of completion rating of 4.0 and long documents a mean rating of 3.0 (Wilcoxon $z = 5.87$, $p < 0.01$). One conjecture is that the additional difficulty of task completion in long documents is due to the large number of participants who used general navigation tools, such as the scrollbar thumb and mousewheel, to complete many of the tasks. Several tasks, such as understanding the document structure, were significantly harder and more time-intensive in longer documents, due to the linear nature of these navigation tools.

The tasks were considered harder to complete in Microsoft Word. The ratings for the two interfaces were significantly different with means of 3.3 for Microsoft Word and 3.7 for Adobe Reader: Wilcoxon $z = 1.9$, $p < 0.05$. It is less obvious why this difference exists; however, it is possibly due to the simpler interface and smaller tool-set available in Adobe Reader. One participant commented that they were overwhelmed by the features that Word provides: “[I] could not find tools efficiently, I knew of features that Word has, but had trouble locating them”.

6.6.2 Freeform Comments

The majority of participants’ freeform comments focused on the tools of which they were unaware or unable to locate. These comments were recorded about Microsoft Word (with possible expert responses in brackets): “zoom has no short-

cut, have to manually set the zoom level” (Ctrl-Mousewheel), “no easy way to quickly jump to a page without scrolling” (goto), “lack of page numbers at bottom” (bottom, left), “Lack of a pages thumbnail view makes finding pages in a large document hard” (View → Thumbnails menu item). As noted earlier, other participants knew that tools existed within Microsoft Word, but were unable to locate them when they were required for use. In contrast, no comments of this nature were recorded about Adobe Reader, the only comment of interest related to the “hiding” of the tools inside the interface: “I forgot about bookmarks but that would probably help also”, validating the empirical evidence that bookmark use increases when the tab is automatically opened, as observed in the longitudinal study.

6.7 Comments from the Interviewees

Following the formal interactive session tasks, participants were encouraged to discuss features they valued in document navigation systems and to air any of their frustrations. They were also asked to comment on document editors and viewers, other than Word and Reader, that they regularly used.

Five areas of discussion arose on multiple occasions. These were: find and search tools, features available in code-editors, inadequacies in zooming techniques, the unpredictability of some tools and general navigation styles.

6.7.1 Find and Search Tools

Several participants commented that they appreciate *type to search* features (for example, those in XEmacs/Emacs and Mozilla Firefox). These tools immediately move to the first match as the user types the letters of a phrase. The primary advantage of this tool is the instant feedback the user receives, allowing the quick identification of a failing search and typing mistakes. The *highlight all* feature in Firefox was also praised, although only two participants applied this feature in Word during the interactive sessions. Participants were keen to see such search features available in Word and Reader.

6.7.2 *Code Editor Features*

All of the participants were computer scientists and spent at least some of their time writing and editing source code. For this task, users employed the following editors: XEmacs/Emacs, TextPad, Microsoft’s Visual Studio, BlueJ and Kate. The navigation features valued in these applications were the type to search features (as above), alphabetic key-based line navigation allowing the hands to remain in the same place on the keyboard (instead of using the arrow keys of the mousewheel), and line and column numbering. Participants were again keen to see some of these tools in their document navigation systems.

6.7.3 *Zoom Tools*

The visual search tasks saw several participants employing zoom tools to view multiple pages simultaneously. However, users were frustrated at the lack of control provided to them, along with sub-optimal page layouts when zoomed out. Most of the zoom controls in these two applications are based on a percent scaling of the original document. However, participants suggested the ability to choose the number of pages to display as a beneficial addition: “I . . . have to guess what zoom level to put it at to see the number of pages I want”. Microsoft Word does allow this in *print preview* mode, but participants wished to perform this action from “within the document”. A further comment was that Reader did not arrange the document’s thumbnails to optimise screen real-estate when zoomed out—instead they are always displayed in one or two columns. Finally, as noted in the *zoom* task, Word also lacked a panning tool. Participants thought this would be particularly useful, especially when using Word’s drop-down box to modify the zoom.

6.7.4 *Tool Unpredictability*

The unpredictability of some navigation tools caused participants to apply caution in their use. One participant commented that they liked the scrollbar because of its “continuity” because jumping to positions was “not always very natural”. A break in this continuity frustrated another user, complaining of the sometimes unpredictable nature of dragging the scrollbar thumb, especially in Reader. Some modes caused the document to jump during or after the scroll action, meaning the user got ‘lost’ after the jump occurred.

The exact functionality of *page-down* features was also questioned—would they replace one screenful of information or would they move to the top of the next page? Because this functionality can differ both within the application (depending on zoom level or page layouts) and between applications, one user was cautious of using it when they required exactly one function or the other.

6.7.5 *Navigation Style*

Finally, participants also made some general comments about their navigation styles. One participant was aware that they generally used “pretty basic ... navigation” and relied heavily on the scrollbar, never employing more advanced features. Another participant, quite validly, questioned whether ending up at the target the fastest was always what was wanted, noting that they employed the scrollbar so that they could get a ‘feel’ for the document content.

6.8 *Limitations*

Three primary limitations exist in these two studies. The first concerns the generalisability of the results—are they specific to these participants or would they apply to all users of document navigation systems? The second is the issue of users modifying their behaviour because they were under study—did this have an impact on the observed results? The third limitation relates to the depth of investigation of each of the tasks—would further probing of a specific task provide a deeper understanding?

6.8.1 *Generalisability*

These two studies have reported the navigation actions and reasons for tool selection from the two participant sets: 37 undergraduate computer science students, and eight computer science postgraduate students and staff. The majority of the task-study participants indicated that they used these applications multiple times a week, if not a day, and the interviewees all rated themselves as *advanced* or *expert* users. There is no reason to believe these results do not generalise to users who fit similar skill levels. Further investigations are required to understand how these patterns differ in beginner users. However, the strong correlation between the re-

sults in both study groups and with the longitudinal study provide some validation that the data collected is generalisable.

6.8.2 The Hawthorne Effect

In both studies, participants were encouraged to perform the tasks as quickly and accurately as possible and in the same manner they would in their everyday work. Hopefully participants followed these instructions. However, like any study that observes human behaviour, there is the possibility that the Hawthorne Effect [147], where users change their actions when they know they are under study, influenced user behaviour. The interactive sessions were most likely to suffer from this effect. Although participants were encouraged to continue as they would everyday, the observations of the interviewer and a video camera must have weighed on some participants' minds. Further, the contextual inquiry techniques employed to extract user thoughts may also have suffered from participants wishing to provide the 'correct' answer. While there is no methodological process for determining the extent of any behaviour changes, the correlations between all studies indicate that it had not significantly influenced the observations.

6.8.3 Study Depth

In this study, participants performed a large number of diverse tasks. However, this meant that individual activities were not studied in great depth. Many of the tasks, for example how an electronic document is read, could form Ph.D. topics in themselves. In this respect, these studies have provided a broad overview of many tasks. The results provide a platform for future studies and work on designing tool improvements.

6.9 Future Work

There are several areas for future work in task-based observations of electronic document navigation. The first two of these address the limitations of the studies presented in this chapter: understanding the actions of users of non-expert skill levels and exploring each of the tasks in more depth. Long duration human-observations or screen-captures of users performing their everyday computing ac-

tivities would also allow a better understanding of how task context influences tool use, the role of document navigation in higher-level tasks and the effect of document familiarity on tool selection.

6.9.1 Users' Skill Levels

Document navigation system users of non-expert skill level are likely to show different patterns of use to those observed in these studies. Long term observations, similar to those described in this and the previous chapter, would allow researchers to understand how a beginner's tool use changes and improves as they become more experienced with the applications and techniques available. This work would also provide valuable insights into the reasons why advanced navigation techniques are not readily adopted, even by expert users.

6.9.2 Task Selection

Task selection for these studies was as broad and encompassing as possible. The disadvantage of such selection is that there was insufficient time to study any one task in great depth. Future studies should consider choosing a smaller number of related tasks and probing these in various contexts and under various conditions. For example, the *answering a question* task would normally be heavily influenced by the user's prior knowledge and emotional connection to the task (will the answer help them to fix their favourite gadget or is the search merely out of interest?).

The tasks presented to users in these studies are likely to only partially contribute to achieving the users' higher-level goals. The context of these goals, the role document navigation plays in higher-level tasks and the user's familiarity with the document can influence the tools selected for use.

6.9.3 In-context Observations

Human-observations or screen-captures of users performing their everyday work would help to understand how document navigation actions contribute to achieving higher-level goals. For example, a user may require the answer to a question to complete a section of a document they are writing. These observations could also provide information on the role and frequency of document navigation when

performing mixed-media tasks, such as comparing a paper and an electronic document, cross-referencing or copy-editing from paper.

These longer studies, combined with more controlled observations could also address the issue of user familiarity with the document at hand. Do users perform search tasks the same or differently in documents that they have written themselves compared to documents they have just downloaded off the Internet? Observations that asked participants to perform tasks in documents that had differing levels of familiarity would also provide useful insights into how tool use differs. This work may also go some way to investigating the value of implicitly gained document knowledge, for example, when scrolling to a section of the document, what else do users learn? What can they tell you about the document after that action?

6.10 Summary

This chapter has presented the results of two task-based observations of electronic document navigation: one asked participants to perform a series of tasks while AppMonitor recorded their actions, the other used a modified think-aloud protocol to understand the reasons participants performed particular actions. Many of the tasks in the two studies were intentionally similar, to allow correlation between the two data-sets.

Overall, the observed behaviours from both studies showed comparable trends, providing a good validator of the collected results, given that the groups of participants were mutually exclusive. Many results also correlated with the data collected during the longitudinal study. One of the most important similarities was users' preference for a small number of tools: many tasks were completed using the scrollbar thumb and the mousewheel. Most participants cited these tools as the easiest or most convenient for task completion. Further probing of the users' knowledge of advanced navigation tools found that users were familiar with a small number, such as find and goto. However, numerous other tools that could aid everyday tasks were unknown to large portions of the participants (such as split-views, document maps and advanced zoom tools). Although many of these techniques are expert tools, most of the participants classed themselves as advanced or expert users. This under-use can partially be attributed to interfaces'

failure to encourage users to employ more advanced tools (as observed in other areas of HCI, for example Grossman et al.'s [88] work).

Coupled with the longitudinal characterisation, it is hoped that these task-based observations will provide a foundation for interaction designers to begin to understand how and why navigation tools are used. Individual analyses will provide a platform for further investigations and the development of improved techniques for efficient task completion.

Part III

Improving Within-document Revisitation

Chapter 7

The Footprints Scrollbar

CHAPTER 5 observed within-document revisitation—the act of returning to a previously viewed document position—is a commonly performed task, but current tools that support this action are rarely used. Chapter 6 found that only some users were aware of revisitation tools and even those users did not always apply them in ‘ideal use’ situations.

Consider the reader of an electronic journal article. She encounters a reference to a paper that she wishes to investigate further. To do so, the reader scrolls her document to the references list at the end of the article, finds the appropriate entry, locates the paper online and then wishes to continue reading from her previous position. Without having accurately attended to the position of the scrollbar thumb, the reader is left with little choice other than to visually scan back through the document to locate her previous reading position.

Revisitation actions, similar to those described in the previous paragraph, are commonly performed. The longitudinal study presented in Chapter 5 observed nearly 9000 occurrences of users returning to previously viewed document locations. Despite the regularity of this action, there was little or no observed use of tools such as bookmarks, split-views and view navigation that can significantly ease this task. The comparison task in Chapter 6 further confirmed that revisitation tools are not employed, even when the task ideally suits their use. Deeper analysis of the longitudinal study log files revealed that users may perform revisitation actions up to 190 times/day. With such high occurrence rates, interaction designers must consider both the length of time users spend completing these tasks and the frustration that users may encounter when they are forced to search for material they have previously viewed.

To aid more efficient completion of revisitation tasks, this chapter proposes a

scrollbar augmentation, called the *Footprints Scrollbar*,¹ that automatically places small marks within the scrollbar trough when a user pauses on a region of the document. These simple marks, combined with mechanisms for quickly returning to the bookmarked positions, make the Footprints Scrollbar significantly faster than a standard scrollbar for performing revisitation tasks.

The remainder of this chapter has the following structure. It begins by summarising the current state-of-the-art revisitation tools. The analysis of revisitation and the use of revisitation tools from the longitudinal study is then reviewed, followed by a brief introduction to the design direction for the Footprints Scrollbar. Next, an analysis of the usefulness of different types of history lists is presented, based on a further log analysis from the longitudinal study. Two experiments that test the performance of adding marks into the scrollbar trough are then described. The design of the Footprints Scrollbar is then presented, based on principles derived from the log analysis and the two previous experiments. This is followed by two further experiments that empirically and subjectively evaluate the Footprints Scrollbar. They compare the Footprints Scrollbar to a standard scrollbar using a controlled lab experiment and a realistic usage study. Finally, this work is summarised and possible directions for future work discussed.

7.1 Summary of Current Revisitation Tools

Current revisitation tools can be divided into two categories: manual tools that require user interaction to mark positions of interest and automatic tools that do not require user intervention to mark locations. These tools were summarised in Section 3.8, but are restated here due to their relevance to the work presented in this chapter.

Manual revisitation tools require an explicit user action to mark information as interesting. For example, bookmark tools in web browsers and Microsoft Word let people explicitly create iconic labels as shortcuts for returning to particular pages or positions in a document. The Bookmark Scrollbar [129] is similar, but bookmarks are placed within a standard scrollbar. However, bookmarks have problems that limit their use in practice [1]. First, they depend on people knowing in advance that the information will be required in the future. Second, people must be-

¹Thanks to Stephen Fitchett for this name.

lieve that adding a bookmark will yield benefits that exceed the manipulation costs of creation and management. Third, people will adopt satisficing strategies [201] (where immediate sub-optimal strategies are favoured over more efficient long-term ones) suggesting that people will often fail to place bookmarks, even when they can foresee the long term advantage [1]. For these reasons, the Footprints Scrollbar provides automatic revisitation support.

Automatic revisitation tools have been developed to support returning actions both between and within documents. Familiar interface controls such as history lists, *Back/Forward* buttons, and *Recent Document* menus facilitate navigation between documents. These automatic revisitation tools minimally intrude on users' activities: they silently record actions, populate a data structure or visualisation and allow revisitation when required. Their primary disadvantages are that people may not understand the algorithm for recording or presenting events and the item set may overwhelm or fail to match the interests of the user. For example, people often misunderstand the behaviour of the web *Back* button, causing frustration when items cannot be revisited [49]. Further, the longitudinal study revealed little use of history mechanisms such as the *Recent Documents* list (see Section 5.10.2).

There are also several widely-deployed examples of automatic revisitation tools for navigating within documents. For example, the web browser's *Back* button works as normal when navigating through internal page links, and Adobe Reader's *Previous/Next View* feature steps through a linear history of scrollbar positions and zoom levels. Microsoft's Visual Studio also has a *previous/next* list of lines that the I-beam cursor has visited. These history lists leave no visible trace in the scrollbar, so people cannot visually scan potential target regions without displaying additional windows or menus.

Finally, Hill et al.'s [99] *Read Wear* system showed a histogram overview of the reading history of an entire document within the scrollbar. Each horizontal line of pixels in the scrollbar encoded information such as the number of edits or length of time reading. Similar scrollbar marks are used by several code editors, but the marks are used to highlight semantic information such as compilation errors rather than to support revisitation.

The concept of document read wear inspired several researchers to examine a variety of techniques for recording and visualising activity beyond the scrollbar. These include Wexelblat and Maes' [227] *Footprints* system, which provided

Tool	App.	Use
1 Split-views	Word	7/13 participants in 2% of docs.
2 User defined bookmarks	Word	Never used
3 Next view	Reader	Never used
4 Previous view	Reader	1/14 participants in two docs. (0.1%)

Table 7.1: Summary of revisitation tool use from the longitudinal study

maps, trails, annotations and signposts for information foraging, and Skopik and Gutwin’s [202] *visit wear* marks for revisitation in fisheye visualisations. The Footprints Scrollbar takes its primary inspiration from Hill et al.’s edit wear and read wear system.

7.2 Summary of Revisitation Log Analysis

The longitudinal study reported in Chapter 5 found that within-document revisitation is a commonly performed task. The full analysis of revisitation, including parameters and limitations, was presented in Section 5.9.3. The pertinent points for this work are briefly summarised in this section.

Within-document revisitation is a commonly performed task. Regions are frequently revisited (Table 5.15, line 2), in a large portion of users’ documents (line 5). On some days, this can occur several hundred times (line 6).

Further, current tools that support revisitation are rarely, if ever, used (Table 7.1). Word’s split-view feature was used by seven participants in 2% of their documents and Reader’s *previous view* was used by one person in two documents. Other tools such as user-defined bookmarks and the *next view* tool were never used during the 120 day study period.

Evidence and motivation for a simple automated revisitation system is drawn from the regularity of revisitation and the lack of use of current tools to support this action.

7.3 Direction for Design

The familiar scrollbar's thumb can be used to help revisitation. For example, a user may know that "moving the thumb roughly four-fifths of the way down will bring me to the *results* section". However, rapid and effective revisitation depends on the user having attended to the thumb's location during previous visits, remembering it, and reproducing it accurately. One or more of these activities can fail.

One way to improve support for revisitation is by augmenting the scrollbar with marks. Such marks are not a new idea. Attribute-mapped scrollbars [233], patented in 1995, used coloured marks in the scrollbar to draw attention to salient properties; scrollbars are well-suited to showing this information, as they provide an overview of the entire document. Hill et al. [99] used a similar approach to denote the read wear that occurs with use.

Read-wear marks show a person locations they have viewed and provide navigation cues to help users quickly return to those positions. However, this idea has not caught on—the author is unaware of any system currently using it (although scrollbar marks are becoming common in IDEs for marking code errors or comments).

The poor adoption of read-wear scrollbars may be due to a lack of knowledge of how revisitation occurs in the real world, how best to design a read-wear scrollbar and the potential benefits and harms of using it in realistic systems. The evidence of revisitation acquired in the longitudinal study is used as motivation to address the knowledge gaps around read-wear based systems and to design and evaluate a new scrollbar to aid users when performing within-document revisitation tasks.

7.4 Effectiveness of Recency and Frequency Based-History Lists for Storing Revisitation Information

The information that describes the document areas a user has visited must be stored in a history list. Displaying all visited positions would likely overwhelm the user, so prioritisation must be given to more important regions. But which regions should be classed as the "more important"?

Hill et al.'s [99] read wear system displayed the amount of time spent on every line of a document, which is useful for returning to frequently used locations, but not for recent locations. This raises the question of whether recency or frequency is a more effective basis for revisitation support.

One key question for either design is that of how many items should be displayed in the scrollbar—the two most recently/frequently visited or the top twenty? Increasing the number of marked items increases the number of positions available for revisitation, but it also increases the user's search load when scanning for a target. The logs from the longitudinal study were analysed (using methodology from previous studies of revisitation [87, 214]) to determine the proportion of revisited items covered by a recency or frequency list of length n .

Figure 7.1a shows the results for recency list analysis. Using this data, a one-item recency list (for example, a simple *Back* button) would allow users to reach 19% and 28% of previous locations with Word and Reader. Longer lists rapidly increase the proportion of accessible locations, with ten-item recency lists covering 81% of revisitations with Word and 84% with Reader.

The frequency list analysis (Figure 7.1b) shows a similarly steep slope, with 10 items allowing access to slightly fewer items than the recency list, at 78% and 83% of revisited locations for Word and Reader respectively. The values for a one-item frequency list are also lower than recency lists at 16% and 13%, meaning that fewer regions would be accessible with a simple *Back* button than with recency lists.

Recency lists are intuitively easier for users to understand and are also more contextually sensitive. If a user shifts from working at the start of a document to the end, the history list will immediately begin to adjust to the newly visited positions. However, with a frequency-based list it may take a period of time and interaction before newly discovered areas of the document are added to the available history list.

With this knowledge, an initial decision was taken to use a recency based history list of ten items. The usefulness of a recency-based system is explored in experiment one; the length of the list is validated further in experiment two (Section 7.6).

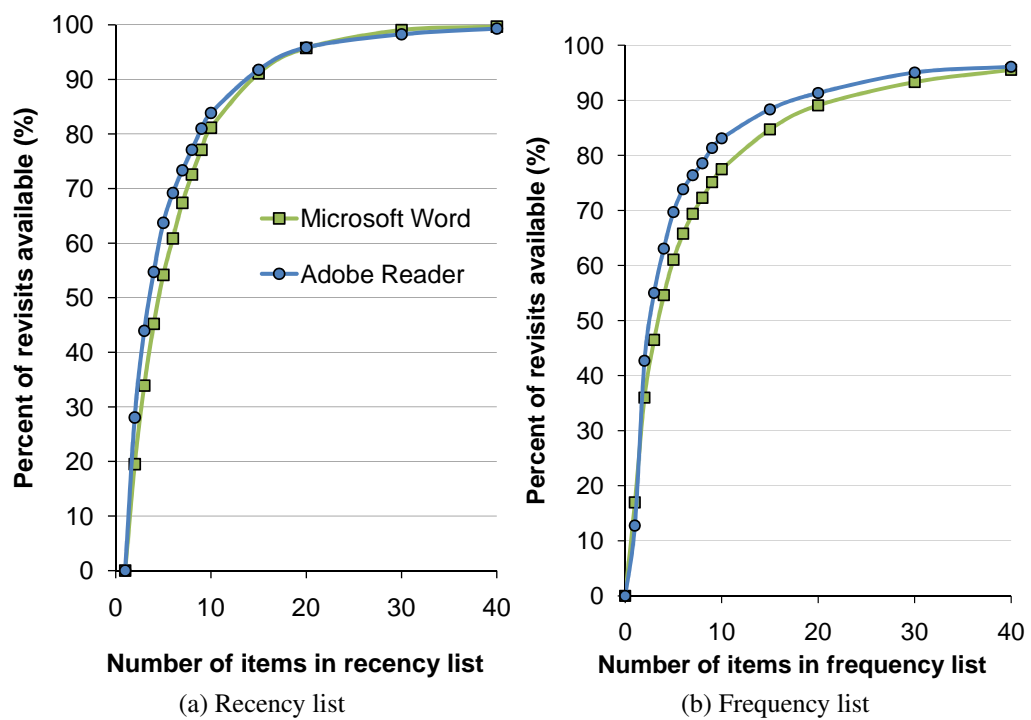


Figure 7.1: Percent of revisited items available through theoretical recency and frequency lists of different lengths

7.5 *Experiment One: Can Recency Marks Help?*

A simple *marking scrollbar* that displayed recency-based marks was designed (Figure 7.2, center) to validate the usefulness of such a system. An initial experiment was conducted to determine whether people would use the recency marks (or ignore them) and whether using them would aid their revisitation performance.

The experimental tasks involved searching for locations in a plain text document (Joyce's *Ulysses* [115]) and periodically revisiting them. This study also investigated how performance was influenced by the number of marks in the scrollbar and by the number of revisits to the same location (the latter providing insight into how well users learn locations).

Users completed the tasks using a simple document-viewing application that only allowed navigation with the scrollbar. Two versions were created, whose interfaces differed only in scrollbar type: a standard scrollbar or a marking scrollbar that showed red marks for visited locations. Middle-clicking on a mark immediately scrolled the view to its associated location.

In this experiment, marks were only placed when targets were successfully visited or revisited; pausing elsewhere in the document did not place a mark. Consequently, this is a best-case study for marking scrollbars; it considers questions about whether people will use the marks, whether the marks improve performance, whether the marks distract from other tasks and how well users learn document locations both with and without marks.

7.5.1 *Participants and Apparatus*

Twelve volunteer university students (seven female) participated in the experiment. Their mean age was 24 years (s.d. 5 years) and they were all experienced users of window-based software (> 10 hours/wk). The experiment lasted approximately 30 minutes. A Java-based system was built for the study. A single window displayed both the experimental interface (Figure 7.2, left side) and a task cueing pane (Figure 7.2, right). The task-cueing pane displayed an initial direction to the next target (up or down), the text of the sentence to be located and a start/finish task button. The direction indicator did not change during the course of a task. The experiment ran on a standard Windows PC with a 22" LCD monitor.

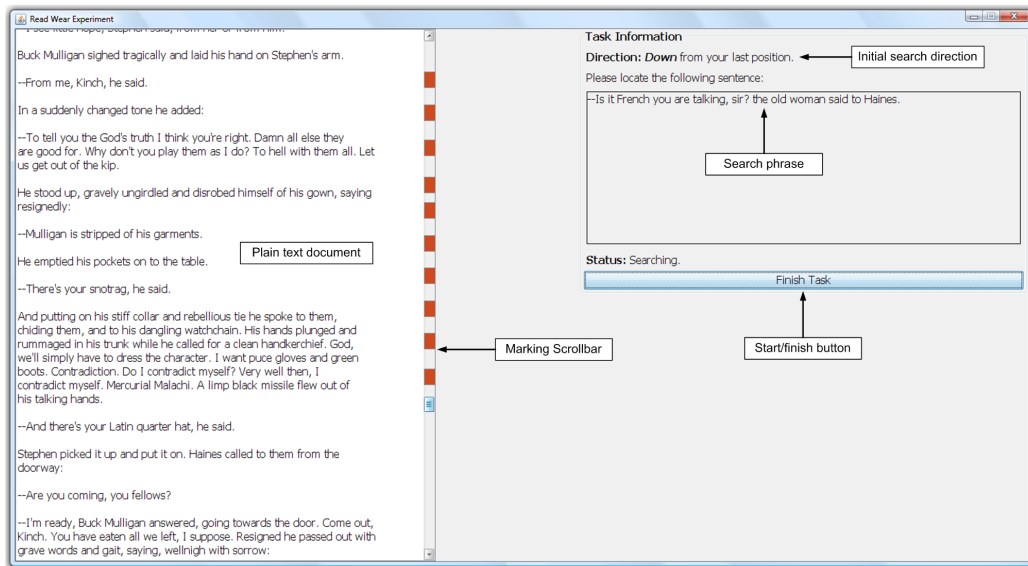


Figure 7.2: Experiment one interface. The *marking scrolling* and a plain text document are shown in the left pane and the task cuing interface in the right.

7.5.2 Experimental Design and Procedure

The task-time dependent measure is analysed using a $2 \times 2 \times 4$ repeated measures analysis of variance (ANOVA), with the three factors: interface type (standard or marking scrollbars), positions visited (5 or 10) and revisit iteration (1st, 2nd, 3rd or 4th revisit).

Participants completed a consent form (Appendix G.1), demographic information sheet (Appendix G.2) and were given a brief introduction to each interface and the experimental method before completing three practice tasks with their first interface. They then completed all tasks with one interface before proceeding to the next, with the order counterbalanced. Participants also completed a NASA Task Load Index (TLX) worksheet [97] (Appendix G.3) and provided comments after completing the tasks with each interface.

Tasks involved using the experimental interface to locate sentences displayed one at a time in the cueing interface. Participants were informed that all target phrases were the first sentences in paragraphs and that tasks were completed by placing the target anywhere within the viewport and clicking the *Finish Task* button. Successfully completing one task automatically cued the next.

Two sets of ten tasks were generated: the first set within the first twenty pages of the document, and the second set within the second twenty pages. All participants used the first set with their first interface. The targets were generated to have similar locations across sets (within one paragraph of each other). Targets were spaced nearly evenly through the 20 page regions, with at least one window of text between them. Equivalent targets in both sets were revisited the same number of times. Participants were unaware of these constraints on target placement.

The set of targets were visited as follows. First, participants found five consecutive targets for the first time, progressing downwards through the document: t_1 , t_2 , t_3 , t_4 , t_5 . Next, they entered the first of two revisitation phases, revisiting items in the order t_2 , t_4 , t_2 , t_1 , t_2 , t_4 , t_2 . They then visited the five remaining targets for the first time (t_6 – t_{10}), resulting in ten marks in the scrollbar, followed by a second revisitation phrase with target order t_3 , t_9 , t_3 , t_6 , t_3 , t_9 , t_3 . In total, each participant completed 48 experimental tasks: two targets visited five times each (t_2 and t_3), two visited three times (t_4 and t_9), two visited twice (t_1 and t_6), and four visited once (t_5 , t_7 , t_8 and t_{10}), giving 24 tasks. These were then repeated for the second interface. Tasks were automatically completed after 90 seconds to reduce the impact of situations where participants became lost. Six tasks were discarded due to exceeding the time limit: two with marks and four with traditional scrollbars.

7.5.3 Results

Figure 7.3 summarises the results from this experiment. The marking scrollbar allowed significantly faster revisitation (mean 11.3 sec, s.d. 11.0 sec) than traditional scrollbars (20.4 sec, s.d. 14.8 sec): $F_{1,11} = 33.8$, $p < 0.001$. As anticipated, there were also significant main effects for revisit iteration ($F_{3,33} = 10.8$, $p < 0.001$), as indicated in Figure 7.3 and for positions visited ($F_{1,11} = 21.2$, $p < 0.005$), with mean acquisition times increasing from 13.1 sec with five visited locations to 18.6 sec with ten. There was a significant interface type \times positions visited interaction ($F_{1,11} = 5.3$, $p < 0.05$), with the marking scrollbar showing greater benefits with more positions visited. There was no interaction between interface type and revisit iteration: $F_{3,33} < 1$, $p = 0.7$.

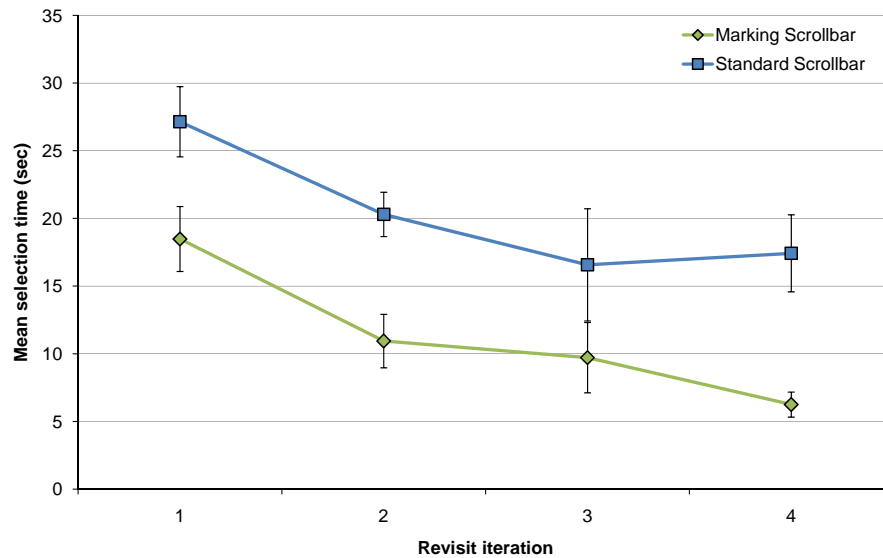


Figure 7.3: Mean target acquisition times in experiment one for the two scrollbar types, across revisit iteration. Error bars show standard error.

Use of the Scrollbar Marks

The direct use of the marks for re-acquiring target positions varied between the participants. Ten participants made heavy use of the middle-click functionality to re-acquire positions. One participant never used the middle-click functionality and one only middle-clicked once. Those participants who did not directly use the marks may have used their visual representation as a guide to potential locations for revisitation.

The marks gave participants discrete options as to the potential locations of the target. One possible tactic for achieving the tasks would be to simply cycle through all of the marks until the correct position was found. Analysis of the experiment's log files showed that participants did not employ this tactic. Instead, they used their spatial memory to become more accurate at target selection as the document positions were visited more frequently. This was demonstrated by a reduction in the mean number of mark selections as a position was more regularly visited.

NASA-TLX	Mean (s.d.)		Significant?
	Standard Scrollbar	Marking Scrollbar	
Mental load	2.9 (1.1)	2.4 (0.9)	0.08
Physical load	2.1 (1.4)	1.7 (0.7)	0.14
Temporal load	2.4 (1.0)	2.3 (1.1)	0.32
Performance	3.3 (1.2)	3.7 (0.9)	0.25
Frustration	2.5 (1.2)	2.2 (1.1)	0.31
Effort	2.9 (0.9)	2.0 (0.9)	0.02

Table 7.2: NASA-TLX responses for standard and marking scrollbars in experiment one. Wilcoxon Signed Ranks significance test.

Subjective Feedback

Subjective measures and participant comments supported these positive results for the marking scrollbar. The NASA-TLX worksheet results (ratings from 1 = low to 5 = high) showed lower mean workload and higher mean performance ratings for the marking scrollbar in all categories, although only the overall *Effort* measure showed a significant effect (see Table 7.2).

The participants' comments were also positive: "the scrollbar mark is extremely nice". One participant also noted that the marks helped to reduce their spatial search space: "I would usually only be off by one mark if I didn't choose correctly the first time, so it was easy to correct my mistakes".

7.5.4 Discussion

These results show that accurately placed scrollbar marks helped participants return to document locations and that the support was appreciated. Also, as more positions were visited, participants gained more from the marks.

There are three questions that are yet to be addressed in this investigation: first, how is performance influenced by large numbers of visited locations (and therefore marks); second, what is the performance of the marking scrollbar when marks are automatically placed wherever the user pauses, rather than the idealised placement on targets tested here; and third, how does the scrollbar work with naturalistic revisitation rather than artificially-organised targets? Experiment two

addresses the first of these questions: how is performance influenced by large numbers of visited locations? Experiments three and four address the later issues.

7.6 Experiment Two: Marking Overload

The log data from the longitudinal study showed that 10 marks can cover over 80% of the revisits users wish to make and that 30 marks can cover close to 100%. As Figure 7.1 reveals, each additional mark covers a smaller proportion of revisitation targets, therefore offering progressively lower utility while increasing the number of distracters. This study investigates how performance is influenced by the number of marks.

7.6.1 Experimental Setup

This experiment used the same marking interface, cuing interface and apparatus as experiment one. In addition, the same people from the first experiment participated in the second, after taking a short break.

Participants completed three practise tasks and then searched and revisited locations within the first 45 pages of Joyce's *Dubliners* [116] using a similar procedure to that used in experiment one (marks were automatically placed on targets when acquired). In this instance, only the marking scrollbar was used, with the number of visited locations (and hence marks) increasing as the experiment progressed.

In a similar manner to experiment one, revisitation time is analysed using a 4×4 repeated measures ANOVA with two factors: positions visited (5, 10, 20 and 30) and revisit iteration (1st, 2nd, 3rd and 4th).

7.6.2 Results and Discussion

The results from experiment two are summarised in Figure 7.4. As anticipated, there were significant main effects for the number of positions visited ($F_{3,33} = 13.4$, $p < 0.001$) and revisit iteration ($F_{3,33} = 22.4$, $p < 0.001$). There is also a significant interaction between the factors ($F_{9,99} = 1.99$, $p < 0.05$), which is best explained by the large absolute task time reduction between iterations one and four with 30 marks, compared to the smaller reductions with fewer marks. This should be ex-

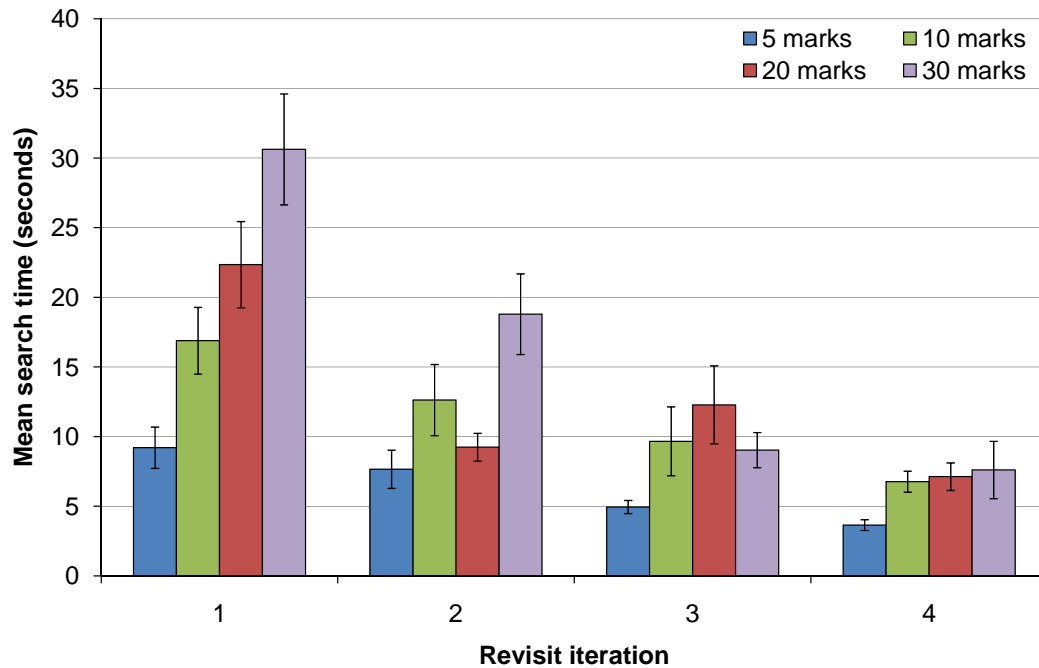


Figure 7.4: Mean revisit times with 5, 10, 20 and 30 scrollbar marks on the 1st to 4th revisit in experiment two. Error bars show standard error.

pected, since learning and remembering thirty marks is clearly more demanding than five.

By the fourth iteration, performance appears to be reaching an asymptote for all conditions, with little performance difference between 10, 20 and 30 marks (Figure 7.4, far right). This suggests that participants were not overloaded by 30 marks. This is further validated by the analysis of the log files that showed more accurate selection of scrollbar marks through each iteration.

These results suggest a tradeoff between the number of marks and performance. Although the participants quickly learned many marks and their associated regions (performing well by the 4th revisit), acquisition times on their first revisit increased steeply with the number of marks (Figure 7.4, far left). Nevertheless, these results show that large mark sets are feasible and that designing to cover nearly all revisits is a possibility. Since ten marks cover more than 80% of revisited locations (Figure 7.1), this value is used in the remaining studies. Real-world implementations of such a system could easily allow a user to configure the

maximum number of marks.

7.7 *The Footprints Scrollbar Design*

The results of the first two studies and the experiences with the marking scrollbar guided the design of a new version of the augmented scrollbar—called the Footprints Scrollbar (Figure 7.5)—that contains a number of more advanced features for supporting revisitation.

The Footprints Scrollbar supports six related methods for revisiting regions. First, coloured marks are placed in the scrollbar to provide spatial cues as to previously visited areas. Marks gradually fade from ‘hot’ colours (reds, oranges) through to ‘cold’ ones (greens, blues) to denote their increasing age in the recency set. Second, middle clicking on a mark causes a rapid, animated scroll transition to the associated view [127]. Third, when the user moves the cursor over the scrollbar, small thumbnail images quickly fade into view alongside each mark, giving a visual overview of the associated document regions. Moving the cursor over a thumbnail expands that thumbnail for better visual inspection (an example is shown at the bottom of Figure 7.5). The thumbnails fade out when the cursor moves away from the scrollbar. Clicking a thumbnail also moves to that view. Fourth, back/forward functions are invoked by the *left* and *right* keyboard arrows; this allows users to rapidly move through the mark history and its corresponding region views. Fifth, depressing the Shift key and rotating the mousewheel moves the document position to the closest mark in the direction of rotation (according to distance, not recency). Finally, marks are overlaid with the numbers 1–10: users can trigger rapid movement to the selected position through the numeric keypad or by typing a number into a *goto* box. Number assignment is arbitrary, with the shortcut remaining constant for the life of the mark. Details of these features follow.

7.7.1 *Marking Algorithms and Behaviour*

Each mark is 16 pixels high (equal to the default minimum size for a scroll thumb in most widget sets) and never overlap one another. Marks are only placed in the scrollbar when the document view remains static for more than two seconds.

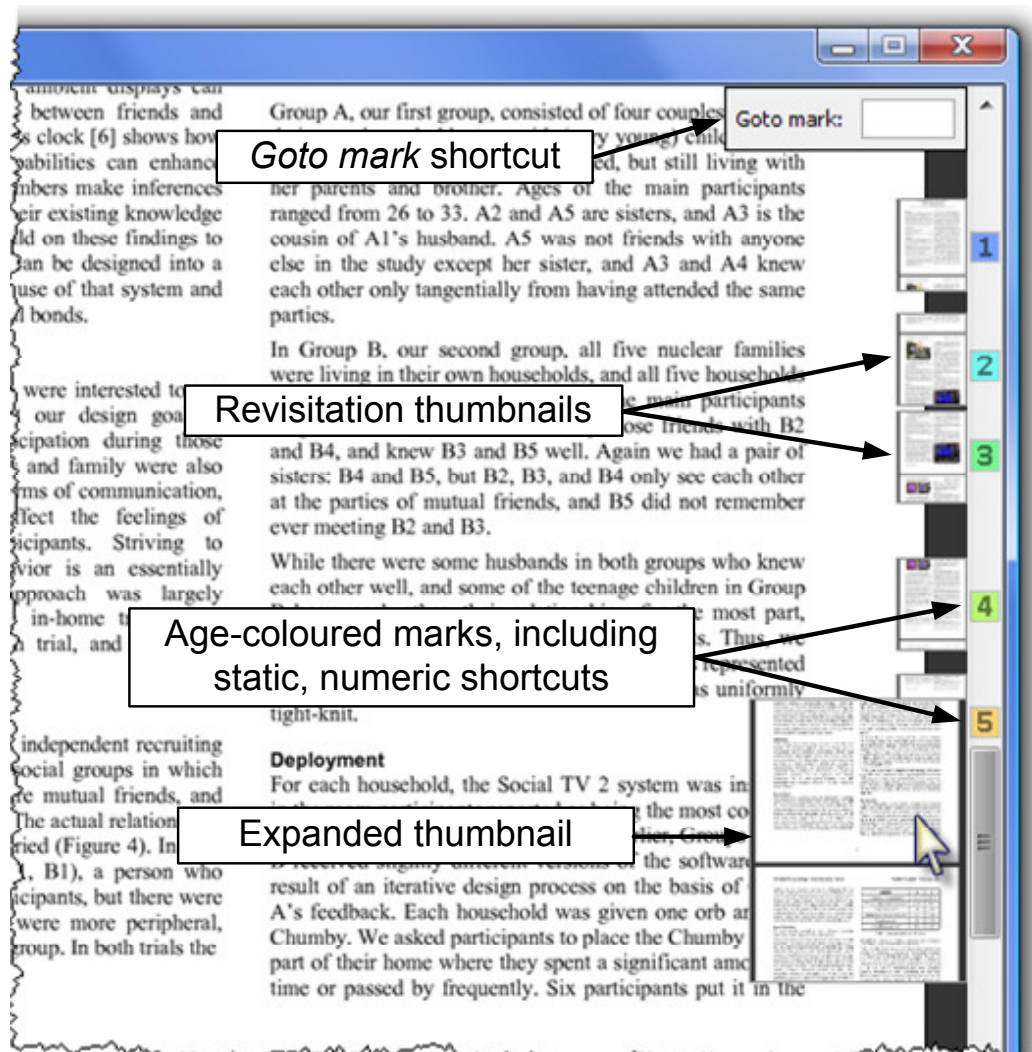


Figure 7.5: The Footprints Scrollbar

Consequently, both continuous scrolling and scrolling with short pauses for device clutching or visual inspection have no impact on the marks and the recency list.

Scrollbar marks are produced from a recency list data structure that removes duplicates [87, 214]. Whenever the view remains static for more than two seconds, that region is inserted at the end of the recency list and any earlier entry for the same location is removed from the list. The current implementation has a 1:1 correspondence between marks and list entries, but this could be relaxed to allow more items on the recency list than are visualised in the scrollbar: for example, providing 10 visible marks, but allowing a larger number of previously visited regions, accessible via the back/forward keys.

Marks are numbered from 1 to n , where n is the configurable maximum (with a default of 10). To reduce visual distraction associated with scrollbar changes, the numbers are stable throughout the life of the mark, with each mark after n receiving the number of the least-recent member of the recency list. Clicking the mark, typing its shortcut number or dragging the thumb to the mark are all equivalent methods of revisitation.

The two-second timeout has an important impact on the behaviour of the interface: it determines when marks are placed and the semantics of the back/forward keys. To help users predict and comprehend this behaviour, the scrollbar thumb depicts the approaching timeout by gradually filling with colour, similar to a progress bar (see Figure 7.6). Once filled, the timeout expires, a mark is left in the scrollbar, the region is inserted at the tail of the recency list and the current view is captured for use in the associated thumbnail.

To precisely describe this behaviour, Table 7.3 uses comma separated letters $a, b, c \dots$ to represent visited document regions; the symbol \perp denotes a two second static location; subscripts $1, 2, 3 \dots$ denote shortcut digits on marks; \Leftarrow and \Rightarrow denote the back and forward keys; and curly braces $\{ \dots \}$ enclose the comma-separated content of the recency list, with the most recent item at the right hand end of the list. An underlined item in the recency list represents the region displayed at the end of each action sequence.

The timeout allows users to dynamically traverse the recency list with the *Back* and *Forward* shortcut keys: for example, quickly flipping between regions (rows 3 and 4 of Table 7.3). However, when editing two document regions (for example, the *Results* section and the *Abstract*) users are likely to spend relatively long pe-

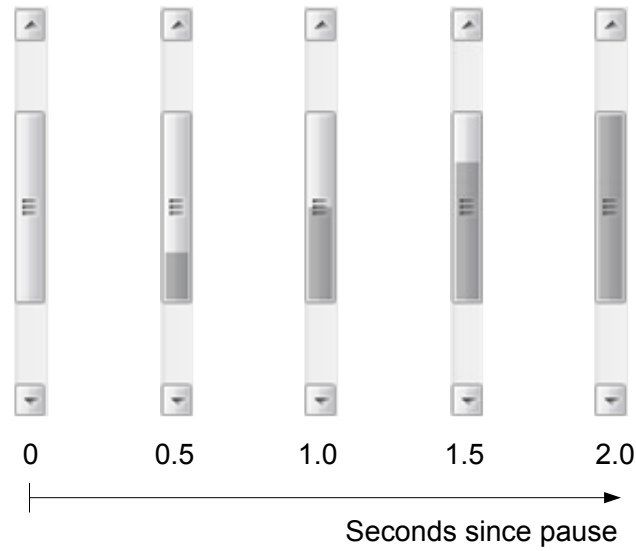


Figure 7.6: Scrollbar thumb-filling animation after a user pauses on a region of the document. A mark is dropped when the two second animation completes.

Actions	Region seq.	Recency list & marks
1 Visit three regions	$a_{\perp}, b_{\perp}, c_{\perp}$	$\{a_1, b_2, \underline{c_3}\}$
2 Scan 3 regions then pause	d, e, f_{\perp}	$\{a_1, b_2, \underline{c_3}, \underline{f_4}\}$
3 'Back' three times (no pauses)	$\Rightarrow, \Rightarrow, \Rightarrow$	$\{\underline{a_1}, b_2, c_3, \underline{f_4}\}$
4 'Forward' twice (still no pauses)	\Rightarrow, \Rightarrow	$\{a_1, b_2, \underline{c_3}, \underline{f_4}\}$
5 Pause	\perp	$\{a_1, b_2, \underline{f_4}, \underline{c_3}\}$
6 'Back' and pause	\Leftarrow_{\perp}	$\{a_1, b_2, c_3, \underline{f_4}\}$
7 'Back' and pause	\Leftarrow_{\perp}	$\{a_1, b_2, \underline{f_4}, \underline{c_3}\}$

Table 7.3: Example navigation behaviour using the Footprints Scrollbar's back/forward history navigation

riods in both areas and they can move between them with subsequent presses of *Back* (rows 6 and 7). Experiment four investigates the usability of this model, and how users chose to interact with the system; first, an experiment that tested the performance of the new system is reported.

7.8 Experiment Three: Footprints Scrollbar Evaluation

The log analysis from the longitudinal study showed that people could benefit from revisitation support and the experimental results described earlier in this chapter suggest that people can benefit when using even a simple marking scrollbar, if marks are correctly placed. However, correct mark placement is an artificial ideal and realistic mark placement needs to be evaluated.

This experiment compared the Footprints Scrollbar with a standard scrollbar in terms of people's performance and preferences. Tasks involved finding and revisiting document regions, with the Footprints system automatically placing marks whenever the user paused for more than two seconds. Unlike experiment one (which controlled the number of revisits to each position), this experiment controlled revisits according to mark position within the recency list. This allows inspection of how the performance of the Footprints scrollbar is affected by the differing ages of marks. The disadvantage of this method is that it artificially made less-recent revisits more likely than the logs show them to be. With the number of positions marked limited to 10 items, this experiment also tests the consequences of the searched position no longer occurring in the marked history list.

7.8.1 Experimental Method

Twelve participants (two female) took part in the experiment after completing an evaluation consent form (Appendix G.4). Tasks involved finding and re-finding document regions that were cued by displaying an image of the target region and an initial direction (up or down, see Figure 7.7, top left). Participants began tasks by pressing a *Start Task* button (center, left), and completed them by scrolling the target region into the middle two-thirds of the screen (right hand side) and clicking a *Finish Task* button. A red status message was displayed if the target was not correctly positioned, and the task continued. Training with each interface was

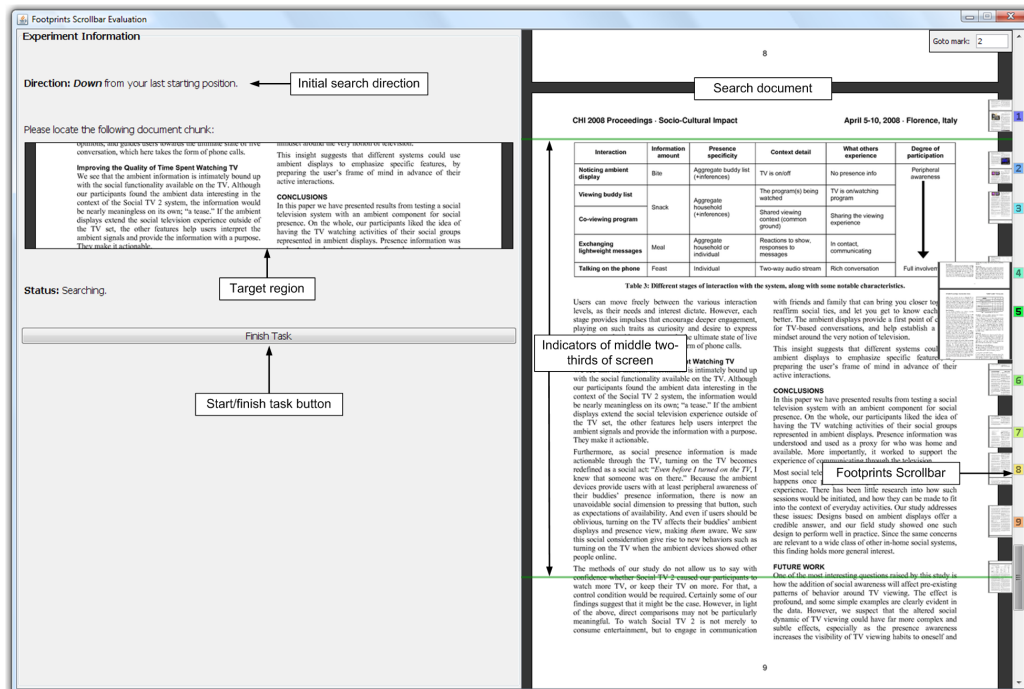


Figure 7.7: Experimental interface for the Footprints Scrollbar evaluation

similar to experiment one and was followed with eight sample tasks to familiarise participants with the procedure.

All participants used the Footprints Scrollbar and the standard scrollbar (in counterbalanced order) with ten- and forty-page documents (two papers from the CHI proceedings and two instruction manuals). Eleven target regions were generated for each document by evenly distributing preliminary locations, then randomly adjusting these locations by between 0 and 5%.

The eleven targets were initially presented consecutively from the top of the document to the bottom. Participants then revisited targets according to their ideal position on the recency list: three times for each position 2–11. The ideal positions assume that marks only fall on targets. However, marks were placed whenever the user paused, so the ideal recency list is likely to differ from the marks on the user's scrollbar. Therefore, latter list positions may not have been visible in the scrollbar when needed, due to being displaced by other marks. Analysis of the study logs showed that the location in the 8th position on the ideal list was in the user's visible

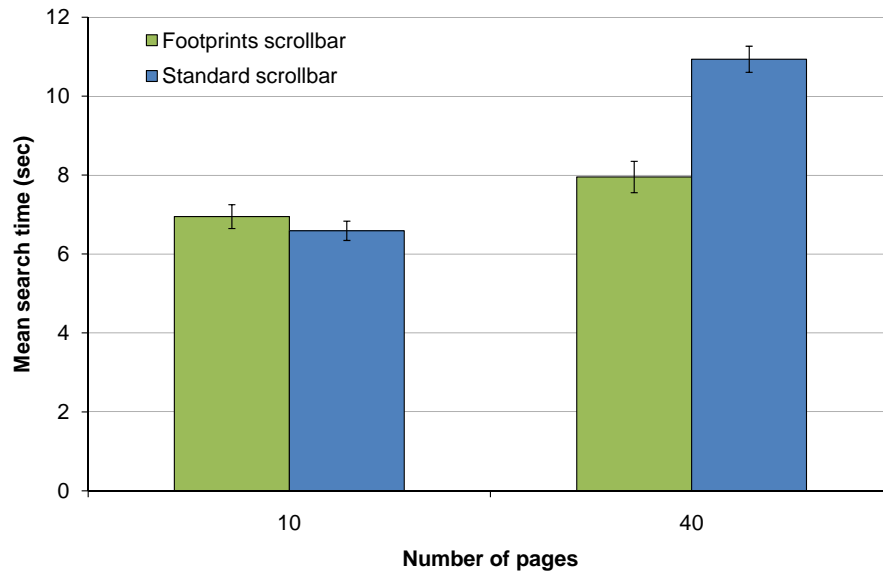


Figure 7.8: Mean revisit times for documents of various lengths in experiment three. Error bars show standard error.

list 90% of the time, the 9th 72%, the 10th 52%, and the 11th 15%.

7.8.2 Results

Task time data are analysed using a $2 \times 2 \times 10$ RM-ANOVA for factors interface (Footprint or standard), document length (10 or 40 pages) and recency list position (2–11).

There was a significant main effect for interface type ($F_{1,11} = 8.24$, $p < 0.05$), with Footprints (7.4 sec) faster than standard scrollbars (8.8 sec). Document length ($F_{1,11} = 69.6$, $p < 0.001$) and recency list position ($F_{1,11} = 18.8$, $p < 0.001$) both showed the expected main effects. Importantly, there were significant type \times length ($F_{1,11} = 15.1$, $p < 0.01$) and type \times position ($F_{10,110} = 2.9$, $p < 0.01$) interactions. Figure 7.8 shows the type \times length interaction—both interfaces performed similarly with 10 page documents (means of 6.9 sec vs. 6.6 sec), but Footprints performed better with 40 page documents (mean 7.9 sec vs. 10.9 sec).

The type \times position interaction is due to the Footprints scrollbar out-performing standard scrollbars in all but the 11th position on the recency list. On trials in the 11th list position, the Footprints system provided a corresponding mark only 15%

of the time (due to marks falling off the end of the recency list), so users who tried to use marks were misled. Means on the 11th trial were 12.0 sec and 10.5 sec for Footprints and standard scrollbars.

7.8.3 Use of Revisitation Tools in the Footprints Scrollbar

The Footprints Scrollbar provides several mechanisms for returning to recently visited document locations. The use (or otherwise) of these tools can provide helpful insights for designers of similar systems for real-world applications.

All participants heavily favoured one revisitation tool when using the Footprints Scrollbar. Four participants only clicked on the shortcut thumbnails. Seven participants used the Shift-Mousewheel technique for an average of 93.0% of their actions (s.d. 12.7%) and one participant primarily used the numeric keyboard shortcuts (72% of their actions). The remainder of participants actions were divided between middle-clicking a mark and using the left keyboard arrow to step back through the history list.

The log data of the use of these tools is consistent with experimenter observations during the study. All techniques were demonstrated to and practised by the participants at the beginning of the study. They were told to use none, some or all of these techniques, as they saw fit, when completing the tasks. However, participants often chose a single technique that they liked and quickly became familiar with that tool while completing the required tasks. The majority of participants commented that they liked a particular tool the most and applied that for many of the tasks.

Subjective Feedback

After completing all tasks with one interface, participants were asked to provide a workload assessment (Appendix G.3) and comment on specific features of the Footprints Scrollbar(APP). All but one participant preferred the Footprints scrollbar; the remaining participant could not choose between the two systems. Mean NASA-TLX worksheet results were uniformly better for Footprints: significantly so for *Physical Load*, *Performance* and *Effort* (see Table 7.4). Participant comments from experiment three are discussed together with those of experiment four in Section 7.10.

NASA-TLX	Mean (s.d.)		Significant?
	Standard Scrollbar	Footprints Scrollbar	
Mental load	3.8 (1.1)	3.3 (0.7)	0.2
Physical load	3.8 (0.6)	2.1 (0.6)	0.01
Temporal load	3.1 (1.2)	2.9 (1.1)	0.3
Performance	2.9 (1.1)	3.7 (0.9)	0.05
Frustration	3.3 (1.3)	2.8 (0.9)	0.1
Effort	3.7 (0.8)	3.0 (0.7)	0.04

Table 7.4: NASA-TLX responses for the standard and the Footprints scrollbars in experiments three. Wilcoxon Signed Ranks significance test.

7.9 Experiment Four: Observational Study

Tasks in experiment three were tightly constrained for experimental control, but they artificially induced revisitation and exaggerated temporally distant revisitations. To inspect more realistic tasks, this study used a structured interview process to observe participants' more natural interaction within a document of their choosing.

Eight participants (one female), all graduate students or faculty from Computer Science, took part in the experiment. They were asked to identify a favourite research paper, which was loaded into the system.

The experiment began with a two-minute introduction to the system's features, explaining marks, thumbnails, digit shortcut keys, and the back/forward arrow keys². Participants then completed 25 tasks using whichever methods they chose. Tasks involved describing the paper to the experimenter in response to a consistent set of questions. Examples of these questions include: "show me what you think is the best part of the paper", "who is on the reference list?", "where is related work summarised?", "what's the first paper referenced in the Introduction" and "where was that paper published" (a complete list can be found in Appendix G.6). Following the interview, participants completed a questionnaire (Appendix G.7) on the effectiveness of various system components. Automatic logs captured all

²For technical reasons, the Shift-Mousewheel technique was not available during this study.

Question	Mean (s.d.)
<i>Scrollbar Marks</i>	
1 Marks were helpful to see where I'd been	4.8 (0.5)
2 Marks were distracting	2.3 (1.2)
3 I understood mark placement	4.9 (0.4)
4 Colours were useful	2.1 (1.0)
5 Marks were useful for revisiting locations	4.9 (0.4)
<i>Mark Shortcuts</i>	
6 Shortcut numbers were useful	2.5 (1.3)
7 Shortcut numbers were distracting	1.5 (0.8)
<i>Back/forward Arrows</i>	
8 Back/Forward keys were useful	2.5 (1.3)
9 I understood the Back/Forward keys	4.5 (1.1)
<i>Thumbnails</i>	
10 Thumbnails were useful for revisiting locations	4.5 (1.1)

Table 7.5: Mean responses to five point Likert scale questions in experiment four

user actions with the interface.

7.9.1 Results

Questionnaire responses (summarised in Table 7.5) show that participants found marks and thumbnails very helpful for visualising and navigating to previously visited locations, with Likert scale (1 = disagree, 5 = agree) means ranging between 4.5 and 4.9 (rows 1–3). These positive ratings were achieved without substantial distraction (mean distraction ratings of 2.3 for marks and 1.5 for shortcut numbers). Overall, comments were highly positive, including “revisitation is tremendously useful and would probably only improve as the document increases in size”, “really useful” and “it’s additive: no interference with any other widget”. Five of the participants stated that they would want all of the supported features in their desktop interfaces; three stated they would want some of the features, with all wanting the thumbnails, region markers and middle-click shortcuts.

Finally, participants were asked to rank the revisitation tools that the Footprints Scrollbar supports (Table 7.6). As was seen in the previous results, the scrollbar

Rank	Tool	Mean rank	s.d.
1	Clicking on marks	2	0.9
2	Clicking on thumbnails	2	1.5
3	Dragging to marks	3.1	1.2
4	Number shortcuts	3.9	1.1
5	Back/forward	4	1.1

Table 7.6: Rankings of the Footprints Scrollbar’s revisitation tools

marks and thumbnails were the most highly favoured, with the back/forward history navigation mechanism ranked the lowest.

Log File Analysis

An analysis of the log files showed that participants, on average, used the provided revisitation tools 21.6 times (s.d. 8.4) during the 25 study tasks. Clicking on thumbnails and middle clicking on the marks were the most popular revisitation methods, accounting for an average of 49% and 42% of the actions respectively.

In a similar manner to that observed in experiment three, five of the participants used predominately one tool—in this case two heavily employed middle clicking on marks and three left clicking on thumbnails. Two further participants split all of their actions between these two tools. One participant applied many of the provided shortcuts.

7.10 Discussion of Experiments Three and Four

Experiments three and four showed the value of the Footprints Scrollbar for revisitation in both a controlled and a more naturalistic situation, especially for longer documents. This success validates the empirical recommendation that revisitation should be supported (log activity analysis) and the initial evaluations of scrollbar marks in artificial situations. People also preferred the features of the Footprints Scrollbar.

Despite these successes, the Footprints scrollbar is still an early design. Although the overall system was praised, participants identified several areas for

improvement, all of which can be incorporated in the next design iteration.

7.10.1 Mark Colours

Nearly all participants commented that colour poorly communicated mark recency. Worse, three participants observed that their memory for items was harmed by colour changes—they might remember the *red* mark in a region, only to be confused by later colour reconfiguration. Furthermore, two participants noted that coloured marks increased the difficulty of visually acquiring the scroll thumb. It is therefore recommended that future implementations use stable mark colours (not denoting mark age) with smaller marks and higher levels of opacity to ensure they do not interfere with scrollbar thumb acquisition.

7.10.2 Digit Shortcuts

Only one of the participants used digit shortcut keys for navigation and another one commented that he used the digit marks “to map locations”. However, the digit marks were criticised by a few participants as either unnecessary or mildly confusing.

7.10.3 Back/Forward Keyboard Shortcuts

Most participants stated that they understood the behaviour of this mechanism (mean 4.5) and that they might use the keys in other documents, but none of the participants actually used them during the tasks. One mentioned a conceptual clash between the ‘Forward’ key and ‘forward = down’ in the document, despite understanding the behaviour. Another stated “using back/forward arrows is something that just didn’t occur to me.” This comment echoes the findings of the log analysis—current recency tools, such as Reader’s *previous/next view* or Word’s bookmarks are not sufficiently ready-to-hand and hence they go unused despite their potential utility.

7.10.4 Lack of Control Over Mark Placement

Finally, some participants felt pressured by the thumb-filling animation, with one commenting that it “made me rush before it dropped a mark” and another stating

that “it would be nice to somehow stop the dropping of the marks”. The questionnaires asked participants to comment on whether the two second marking timeout was too short or too long, with several responding that it was “sometimes too short and sometimes too long”. Lightweight controls for manually adding and removing marks in the scrollbar could solve this problem, but it is unclear whether people would use such controls.

7.11 *Summary and Future Work*

Revisitation has been comprehensively investigated in domains such as web navigation and command use. Somewhat surprisingly, region revisitation within documents has been largely overlooked. The longitudinal study’s log analysis demonstrated that users frequently revisit document regions and that short revisitation lists can theoretically provide access to most locations that users return to. These findings are used to motivate and inform the design of a system, based on Hill et al.’s read wear, that augments the scrollbar with marks that aid revisitation; a series of evaluations have demonstrated that this system can improve user performance.

The Footprints Scrollbar works within the current ecology of graphical user interfaces—it augments the familiar scrollbar rather than replacing it and it occupies the same location and screen-area. Except for shortcut keys (which can be easily modified) the input to this scrollbar does not compete with input actions that control other parts of a document viewer or GUI. While improvements were suggested through experimental feedback, these can be easily incorporated in future system refinements.

This research into the Footprints Scrollbar provides several avenues for future work. First, the design of the scrollbar can be revised, according to the users’ responses and a version produced that can be incorporated into real-world document readers. Second, longer-term evaluations of the tool need to be performed, to provide additional information about how revisitation and revisitation support work in real use. This incorporation into a real-world document viewer would also allow hypotheses regarding the usefulness of maintaining mark information across document sessions to be tested. Finally, other domains, such as code-editors and spreadsheet applications, may also benefit from revisitation aids such as the Foot-

prints Scrollbar. This work should begin with an investigation as to the extent of revisitation within these domains.

Part IV

Discussion and Future Directions

Chapter 8

Discussion and Future Work

ELECTRONIC document navigation is a commonly performed activity that allows users to view information spaces too large to conveniently be displayed on their screen. It is an action conducted not only in dedicated document readers, such as those discussed in this thesis, but in any application that has content too large to be displayed in its entirety. These applications display documents such as email messages, web-pages, spreadsheets, even images and music playlists. The work presented in this thesis provides researchers with an empirical foundation, complemented with reasons for tool selection, to guide the redesign of within-document navigation tools. It also studied one commonly performed activity—within-document revisitation—and designed the Footprints Scrollbar to better support this task.

The characterisations presented in this thesis are by no-means complete. Other researchers can use the analysis and results presented here as a platform for further characterisations, improvements to current tools and a starting point for comprehensively modelling users' navigation behaviour. Even studies run in an identical manner to those described here would be beneficial to the research community, providing further validation of the collected data.

This chapter takes a higher level view of this work, it describes the progress on the research objectives, presents lessons for practitioners, discusses the generalisability of the results, looks at the current directions of document navigation and provides suggestions for areas of future work.

8.1 Progress on Research Objectives

The over-arching goal of this thesis was to understand and improve electronic within-document navigation. More specifically, the thesis set out to achieve four

high level goals:

1. Understand the tasks that require document navigation and the electronic document navigation tools currently available to complete these tasks.
2. Create an empirical characterisation of *real world* electronic document navigation.
3. Understand the reasons users choose to employ particular document navigation tools and examine the limits of user knowledge of currently available techniques.
4. Analyse, design and evaluate a new tool to aid a commonly performed navigation task: within-document revisitation.

When defining these goals, Chapter 1 also outlined criteria for judging their successful completion. The alignment of the research outputs with these criteria are now discussed.

Objective 1 was deemed to be complete when both document navigation tasks and electronic document navigation tools were identified, described and categorised. The first part of this goal, understanding navigation tasks, was completed via a literature review of previous work into paper and electronic document navigation. Chapter 2 identified and presented the five task groups: ‘overviewing and browsing’, ‘reading’, ‘writing and annotation’, ‘search’, and ‘revisitation’. The second part of Objective 1, investigating the tools available for electronic document navigation, was also successful in the creation of a tool classification. Chapter 3 reported eight navigation tool categories that encompassed current techniques: core document navigation tools, input devices, scrollbar augmentations, content-aware navigation aids, document visualisations that provide multiple views, navigation by indirect manipulation, zoom tools and revisitation tools.

Objective 2 required a characterisation that described all facets of document navigation and classified or generalised user actions. To aid completion of this goal, AppMonitor was implemented to allow user actions in Microsoft Word and

Adobe Reader to be unobtrusively logged. Using this tool, the document navigation actions of 14 participants were monitored over 120 days. Chapter 5 documented the results of this longitudinal study. A large range of navigation statistics were reported including the length of time documents were open, a characterisation of vertical document navigation actions, the percent of users' time spent navigating and the use of zooming, page layout and window management tools. Users were classified by placing them into stereotypical navigator categories. More general conclusions regarding navigation were also drawn: a small number of tools are used for the majority of navigation and advanced navigation tools are rarely employed.

Objective 3 required an explanation of the observations from the longitudinal study. It was to also provide an analysis of the limits of user knowledge of currently available navigation techniques. Chapter 6 presented two task-based observations of electronic document navigation. The first asked participants to perform specific navigation tasks while AppMonitor recorded their actions. In the second, participants took part in interactive sessions to understand the reasons for selecting particular tools for specific tasks. These observations covered 14 tasks, from all areas of the navigation task categories documented in Chapter 2. The observation of little use of advanced navigation tools was primarily explained by users' lack of knowledge of these techniques.

Objective 4 was deemed successful if the newly designed revisitation tool was significantly faster and subjectively preferred over a standard scrollbar. Chapter 7 presented the analysis, design and evaluation of the Footprints Scrollbar, a tool that places marks in the scrollbar trough to aid users when returning to recently visited positions within their documents. The empirical evaluations found the Footprints Scrollbar to be significantly faster for revisitation tasks and to be subjectively preferred by participants.

Finally, in line with the overall research objective, this thesis has presented characterisations that have significantly improved the research community's knowledge of within-document navigation. The value of this knowledge to improving navigation was demonstrated in the design of the Footprints Scrollbar.

8.2 Lessons for Practitioners

The discussion and summaries of individual chapters have offered guidance to practitioners continuing this work or employing the results. Three further lessons can also be offered:

1. *A small number of navigation tools are used for a large number of tasks.* After practise, users become confident and accurate with a small number of tools: the scrollbar, the mousewheel and paging tools. These tools are employed in a large percentage of situations, even if they are not the most efficient for the task at hand. This observation is a further demonstration of Zipf's law [237], the Pareto principle [117] and the '80–20 rule', each of which describe the human behaviour of regularly choosing a small number of favoured options from a much larger tool-set. Designers of new applications that display navigable data should, at a minimum, support all three of these tools.
2. *Users would benefit from 'ready to hand' education that advances them from beginner to expert users.* Many advanced navigation tools are, by default, 'hidden' from the user (for example, Reader's bookmarks). These tools must compete with the generic scrollbar which is always visible. Apart from system documentation (which is not used [40]), current applications make little or no effort to inform and educate users of more efficient mechanisms of achieving tasks. Without such assistance, it is unreasonable to expect users to simply evolve from beginners to experts.
3. *Observation of current use informs future design.* The data recorded by AppMonitor in the longitudinal study and the observations in the task-centric study were used to inform the design of the Footprints Scrollbar. Without these observations, it is unlikely that designers would be aware that revisitation is a commonly performed action and the tools provided to support it are rarely used. The owners of Customer Experience Programs that already collect such data (for example, Microsoft [157] and Adobe [6]) should strongly be encouraged to disseminate the results to the research community. This

lesson applies not only to document navigation, but also more generally to HCI.

8.3 Research Generalisability

The research presented in this thesis has focused on within-document navigation in dedicated document viewers. However, this work is generalisable and extendable both within and beyond this domain. This section outlines three such areas: the use of AppMonitor to record user actions in any Windows program, applying the characterisations of document navigation beyond the two studied systems and extending document revisitation to other domains.

8.3.1 AppMonitor

AppMonitor was employed to record document navigation actions in Microsoft Word and Adobe Reader. However, AppMonitor's architecture readily allows extension to monitor other applications. For general event recording (such as mouse events, keyboard events, menu selections and button presses) only small code changes would be required. To acquire a deeper understanding of the studied system's use, researchers may wish to further extend AppMonitor to record important application content or context.

In this research, AppMonitor incorporated several features specific to document navigation—determining the length of a document, and recording changes to the scrollbar's position and to the document's zoom. Similar domain specific information could also be captured in other applications—for example, in an Integrated Development Environment (IDE), researchers may be interested in observing the number of errors and warnings after each compilation. Chapter 4 provides further details on the changes required to extend AppMonitor.

8.3.2 Characterisations of Document Navigation

The document navigation characterisations presented in this thesis describe the actions of the users who participated in these studies. However, there is no reason to believe that similar patterns of use do not exist for users of similar skill levels—in this case, mainly self-rated advanced or expert users.

Further, because the majority of tool use was with three core navigation tools (the scrollbar, the mousewheel and paging tools) these results may also generalise to applications other than dedicated document viewers. It would not be unreasonable to hypothesise that similar use of these navigation tools would be present in applications such as spreadsheets and web-browsers. One explanation for the high use of these tools is *because* they appear in a large number of applications. It is also expected that other domains that require navigation would observe similar satisficing tool selection, with advanced tools under-utilised.

Many of these application areas have not had their navigation mechanisms studied in detail. Research that identified differences or commonalities with this work would be valuable for high-level interaction designers. This would allow the tools provided in operating system level toolkits to better cater for the needs of users in many applications.

8.3.3 Revisitation and the Footprints Scrollbar

The Footprints Scrollbar was designed to aid users more efficiently perform within-document revisitation tasks. It was evaluated in the setting of a viewer similar to Microsoft Word or Adobe Reader—one that could view documents such as conference papers and product manuals.

Previous research has observed revisitation in a large number of other computer-based tasks [87, 179]. Indeed, Hill et al. [99] also suggested their read wear concept for spreadsheets and menus. Because the Footprints Scrollbar only provides additional features to the scrollbar and does not reduce or remove any of the current functionality, it is an ideal candidate for deployment to other applications such as code editors, web-browsers or spreadsheets.

For other forms of data where small thumbnails do not satisfactorily represent the content, techniques such as semantic zooming (see Section 3.7.1) could be applied to highlight pertinent areas of the marked view.

8.4 Document Navigation into the Future

Documents have existed for hundreds of years and are unlikely to disappear anytime in the foreseeable future. In contrast, it would be a reasonable assumption that their use will continue to increase with the ever-growing popularity of the

Internet as a place to publish any and every document. Their widespread availability should provide navigation researchers with further motivation to improve document navigation tools.

Traditional desktop-based document navigation systems, such as Microsoft Word and Adobe Reader, are now facing competition from online competitors. Google Docs [85] allows users to create documents online through a web-browser. Its PDF reader also allows PDF documents to be viewed within a web-browser, without requiring additional client-side software. Moving from desktop-based to online systems has both advantages and disadvantages for document navigation. Developers will very quickly be able to deploy improvements to users, much faster than a typical cycle of development, shipping and purchasing of office suites or even downloading of upgraded versions. However, while Internet bandwidth is ever-increasing, it will be a long time before web-based document navigation systems can stream graphics-intensive animations and provide the fluid interactions available in desktop-based systems. This inevitably means that the online systems will be less fully-featured than desktop systems, often at the expense of the more complex, but time-saving, navigation mechanisms.

Document viewing and editing is also becoming more mobile—e-book readers and small screen mobile devices can display and provide editing facilities away from the desktop computer. These devices suffer from the same navigation issue as on a desktop—what is the best way to perform a particular navigation task in a particular document? However, they also pose additional challenges, such as less processing power (due to battery requirements), smaller display space and different methods of interaction (with no mouse or keyboard available). While it seems unlikely that small screen mobile devices without keyboards would be used for large scale document creation, it is probable that an ever-increasing number of documents will be read on such devices, providing researchers with even further document navigation challenges.

8.5 Future Work

Desktop-based electronic within-document navigation is far from a solved problem. This thesis has provided, to the best of the author's knowledge, the first longitudinal characterisation of electronic within-document navigation using com-

monly available *unmodified* applications. The remainder of this section suggests possible areas for future work.

8.5.1 Understanding the Context of User Actions

The longitudinal study provided an analysis of the specific navigation actions that users conducted and the task-centric observations presented insights into the reasons for tool selection. A further study could expand this work to understand navigation actions in the context of users' overall objectives—for example, understanding what the user wished to achieve when they scrolled down two pages in a document.

The task-centric studies provided users with a constrained series of tasks to examine tool selections. However, navigation is not performed in such isolation and instead is used to aid the completion of a higher-level objective. To provide context for this navigation, human-observations and screen-captures of users' actions would provide insights into how navigation contributes to them achieving their higher-level goals. A study of this nature could easily include an investigation into the documents used and their content. Researchers would then understand the order and frequency of occurrence of particular goals and whether navigation mechanisms are consistently applied when completing them. Models of user interactions in specific contexts could then be created.

8.5.2 Modelling User Actions

The studies presented here have provided characterisations of the use of navigation tools. However, they do not provide models for predicting tool use for particular tasks. By understanding the content of the documents and the situations where particular types of navigation are employed, researchers can begin to create models for particular groups of users. These models would allow researchers to provide training or tools specific to their users' needs.

8.5.3 Development of Interaction Techniques

The characterisations presented in this thesis have provided insights into how and why document navigation tools are employed. Interaction designers can use many

aspects of these characterisations, possibly aided by future user modelling, to develop new interaction techniques.

This work selected one specific navigation interaction—revisiting document regions—for in-depth study and development of a tool to aid this task. Future developments may include encouraging the use of advanced navigation tools and improving within-document search features.

The development of new tools in this area is strongly encouraged; however, designers should be cautious to not overload users with too many navigation tools. Several of the studies in this thesis have documented the small number of tools applied in most navigation situations. One reason for this may be that current users are overwhelmed when choosing a tool. The addition of a new tool to an application should be carefully considered and complimented with mechanisms for educating the intended audience on its presence and use.

8.5.4 *A Standardised Evaluation Framework*

Document navigation researchers would benefit from a standardised methodology for evaluating newly developed tools, as is available in other areas of HCI. Mackay et al.'s [140] *Touchstone* framework provides a generalised architecture for aiding the setup of experiments. However, frameworks of this nature must still be supported by standardised evaluation methodologies, such as the ISO9241 [66] standard that specifies the evaluation conditions for Fitts' law [74] experiments. All document navigation evaluations are currently performed in an 'ad hoc' manner, with individual researchers selecting documents, tasks and conditions that they believe to be a fair evaluation of their system. Unfortunately, this makes it difficult for later comparison of tools without reproduction of the original evaluation.

A document navigation evaluation framework would consist of standardised documents (covering the spectrum of those widely viewed), common tasks that can be used to evaluate particular activities and standard conditions for performing those activities (including a methodology for cuing tasks).

A corpus of documents, while never fully inclusive, would at least allow experimenters to employ common documents in their assessments. Attributes to be considered when creating this document set include the length of a document, its

layout (vertical vs. horizontal, continuous vs. discrete pages, presentation size) and the content (percent and size of text, percent of images and white-space, and the organisation of the content). Researchers should be aware that ‘standard’ document types and lengths are likely to evolve over time and such a corpus would need regular updates.

Tasks in this framework would be drawn from a wide variety of sources, including previously performed studies, those described in this thesis and any additional tasks observed in future studies of document-based tasks. A methodology for providing tasks to users and standards for communicating task completion should also be included. Finally, researchers may wish to consider a measure of user familiarity with a document—a factor that can influence navigation tool selection.

This framework and methodology guidelines would allow fairer comparison and easier reproducibility of experiments, both within and between researchers.

Chapter 9

Conclusion

THIS thesis was motivated by the knowledge gap researchers faced when developing new within-document navigation tools—they had little evidence of the actions users currently perform. In an attempt to remedy this situation, this thesis has investigated how electronic document navigation tools are employed in users' everyday work environments and explored reasons for this tool use through two task-centric studies. It then took one commonly observed behaviour, within-document revisitation, and presented the analysis, design and evaluation of a Footprints Scrollbar to aid more efficient completion of this task.

This thesis has made five primary contributions to the research knowledge in the domain of electronic document navigation. These are:

1. A review of electronic document navigation tools. Prior to this research, no such review that brought all within-document navigation tools together existed. This review allows other researchers to more quickly analyse the existence of current tools and opportunities for future developments.
2. The development of AppMonitor, a tool for recording user actions in unmodified Windows applications. Before this work, researchers had no automated tool that could collect data pertinent to electronic document navigation, unobtrusively, in *unmodified, real world* applications. Further, this tool readily allows for extension to log user actions in any Windows based application.
3. An empirical characterisation of electronic document navigation. Using AppMonitor, 14 participants' document navigation actions were recorded over a period of 120 days. This characterisation describes the use and re-use of documents, the use of vertical and horizontal navigation tools, the

percent of users' time spent navigating and the use of zooming, page layout and window management tools. User actions were classified by placing them into stereotypical navigator categories. Overall, it found that three tools (the scrollbar thumb, the mousewheel and the paging tools) were used for the majority of navigation and that advanced techniques were rarely, if ever, employed.

4. Two task-centric studies that provide reasons for navigation tool selection. The first asked 37 participants to perform a series of constrained tasks in Microsoft Word and Adobe Reader. The second consisted of eight interactive sessions that asked users to complete a series of tasks and then provide the experimenter with justifications for their tool choices. These studies also provide insights into users' knowledge of advanced navigation techniques.
5. The analysis, design and implementation of a Footprints Scrollbar that aids within-document revisitation tasks. This system places small marks inside the scrollbar trough when a user pauses on positions within the document. It provides thumbnail previews of these regions and a range of methods for quickly returning to the previously visited positions. The Footprints Scrollbar was significantly faster and subjectively preferred over a standard scrollbar for revisitation tasks.

This research provides the foundation for further work in this area. Researchers can use these findings as a platform for future observations of document navigation and to begin modelling tool use in this domain. Interaction designers can use these observations to inform the design of the next generation of document navigation tools.

References

- [1] Abrams, D, Baecker, R, and Chignell, M. Information Archiving With Bookmarks: Personal Web Space Construction and Organization. In *CHI '98: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, California, United States, 1998. ACM Press/Addison-Wesley Publishing Co., pages 41–48. ISBN: 0-201-30987-4. doi: 10.1145/274644.274651.
- [2] Adar, E, Teevan, J, and Dumais, S. T. Large Scale Analysis of Web Re-visitation Patterns. In *CHI '08: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 2008. ACM, pages 1197–1206. ISBN: 978-1-60558-011-1. doi: 10.1145/1357054.1357241.
- [3] Adler, A, Gujar, A, Harrison, B. L, O'Hara, K, and Sellen, A. A Diary Study of Work-Related Reading: Design Implications for Digital Reading Devices. In *CHI '98: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, California, United States, 1998. ACM Press/Addison-Wesley Publishing Co., pages 241–248. ISBN: 0-201-30987-4. doi: 10.1145/274644.274679.
- [4] Adobe Systems Incorporated. *Document Management—Portable Document Format—Part 1: PDF 1.7*, first edition, 2008. Available online: http://www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf, last accessed 5 December 2008.
- [5] Adobe Systems Incorporated. Adobe Reader. Available online: <http://www.adobe.com/products/reader/>, last accessed 29 April 2009.
- [6] Adobe Systems Incorporated. Adobe Product Improvement Program. Available online: <http://www.adobe.com/misc/apipfaq.html>, last accessed 7 October 2008.

- [7] Ahlberg, C and Shneiderman, B. The Alphaslider: A Compact and Rapid Selector. In *CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Boston, Massachusetts, United States, 1994. ACM, pages 365–371. ISBN: 0-89791-650-6. doi: 10.1145/191666.191790.
- [8] Aliakseyeu, D, Irani, P, Lucero, A, and Subramanian, S. Multi-Flick: An Evaluation of Flick-Based Scrolling Techniques for Pen Interfaces. In *CHI '08: Proceeding of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 2008. ACM, pages 1689–1698. ISBN: 978-1-60558-011-1. doi: 10.1145/1357054.1357319.
- [9] Amazon.com Inc. Amazon.com Kindle: Amazon’s Wireless Reading Device. Available online: <http://www.amazon.com/kindle/>, last accessed 20 November 2008.
- [10] Andersen, T. H. A Simple Movement Time Model for Scrolling. In *CHI '05: Extended Abstracts on Human Factors in Computing Systems*, Portland, OR, USA, 2005. ACM, pages 1180–1183. ISBN: 1-59593-002-7. doi: 10.1145/1056808.1056871.
- [11] Appert, C and Fekete, J.-D. OrthoZoom Scroller: 1D Multi-Scale Navigation. In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006. ACM, pages 21–30. ISBN: 1-59593-372-7. doi: 10.1145/1124772.1124776.
- [12] Apple Inc. Mighty Mouse, 2008. Available online: <http://www.apple.com/mightymouse/>, last accessed 8 October 2008.
- [13] Apple Inc. Mac OS X Leopard, 2008. Available online: <http://www.apple.com/macosx/features/300.html#spotlight>, last accessed 26 October 2008.
- [14] Apple Inc. Apple - iPhone, 2008. Available online: <http://www.apple.com/iphone/>, last accessed 20 November 2008.

- [15] Apple Inc. Apple Portables: Two Finger Trackpad Scrolling, 2009. Available online: <http://support.apple.com/kb/HT3448>, last accessed 13 May 2009.
- [16] Apple Inc. Apple Human Interface Guidelines: Window Behavior, 2008. Available online: http://developer.apple.com/documentation/UserExperience/Conceptual/AppleHIGuidelines/XHIGWindows/chapter_18_section_5.html#//apple_ref/doc/uid/20000961-TPXREF26, last accessed 22 October 2008.
- [17] Arai, T, Aust, D, and Hudson, S. E. PaperLink: A Technique for Hyperlinking from Real Paper to Electronic Content. In *CHI '97: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Atlanta, Georgia, United States, 1997. ACM, pages 327–334. ISBN: 0-89791-802-9. doi: 10.1145/258549.258782.
- [18] Atterer, R and Schmidt, A. Tracking the Interaction of Users with AJAX Applications for Usability Testing. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007. ACM, pages 1347–1350. ISBN: 978-1-59593-593-9. doi: 10.1145/1240624.1240828.
- [19] Atterer, R, Wnuk, M, and Schmidt, A. Knowing the User's Every Move: User Activity Tracking for Website Usability Evaluation and Implicit Interaction. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, Edinburgh, Scotland, 2006. ACM, pages 203–212. ISBN: 1-59593-323-9. doi: 10.1145/1135777.1135811.
- [20] Baecker, R. M, Nastos, D, Posner, I. R, and Mawby, K. L. The User-Centered Iterative Design of Collaborative Writing Software. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993. ACM, pages 399–405. ISBN: 0-89791-575-5. doi: 10.1145/169059.169312.
- [21] Bartlett, J. F. Rock 'n' Scroll Is Here to Stay. *IEEE Computer Graphics*

- and Applications*, 20(3):40–45, 2000. ISSN: 0272-1716. doi: 10.1109/38.844371.
- [22] Baudisch, P, Lee, B, and Hanna, L. Fishnet, A Fisheye Web Browser with Search Term Popouts: A Comparative Evaluation with Overview and Linear View. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 2004. ACM, pages 133–140. ISBN: 1-58113-867-9. doi: 10.1145/989863.989883.
 - [23] Bederson, B. B and Hollan, J. D. Pad++: A Zooming Graphical Interface for Exploring Alternate Interface Physics. In *UIST '94: Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology*, Marina del Rey, California, United States, 1994. ACM, pages 17–26. ISBN: 0-89791-657-3. doi: 10.1145/192426.192435.
 - [24] Beeching, W. A. *Century of the Typewriter*. Heinemann: London, 1974.
 - [25] Björk, S. Hierarchical Flip Zooming: Enabling Parallel Exploration Of Hierarchical Visualizations. In *AVI '00: Proceedings of the Working Conference on Advanced Visual Interfaces*, Palermo, Italy, 2000. ACM, pages 232–237. ISBN: 1-58113-252-2. doi: 10.1145/345513.345324.
 - [26] Björk, S and Redström, J. An Alternative to Scrollbars on Small Screens. In *CHI '99: Extended Abstracts on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, 1999. ACM, pages 316–317. ISBN: 1-58113-158-5. doi: 10.1145/632716.632908.
 - [27] Boguraev, B, Kennedy, C, Bellamy, R, Brawer, S, Wong, Y. Y, and Swartz, J. Dynamic Presentation of Document Content for Rapid On-Line Skimming. In *AAAI: Spring 1998 Symposium on Intelligent Text Summarization*, Stanford, CA, 1998. pages 109–121.
 - [28] Bourgeois, F, Guiard, Y, and Lafon, M. B. Pan-Zoom Coordination in Multi-Scale Pointing. In *CHI '01: Extended Abstracts on Human Factors in Computing Systems*, Seattle, Washington, 2001. ACM, pages 157–158. ISBN: 1-58113-340-5. doi: 10.1145/634067.634163.

- [29] Bourgeois, F and Guiard, Y. Multiscale Pointing: Facilitating Pan-Zoom Coordination. In *CHI '02: Extended Abstracts on Human Factors in Computing Systems*, Minneapolis, Minnesota, USA, 2002. ACM, pages 758–759. ISBN: 1-58113-454-1. doi: 10.1145/506443.506583.
- [30] Brewster, S. A, Wright, P. C, and Edwards, A. D. N. The Design and Evaluation of an Auditory-Enhanced Scrollbar. In *CHI '94: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Boston, Massachusetts, United States, 1994. ACM, pages 173–179. ISBN: 0-89791-650-6. doi: 10.1145/191666.191733.
- [31] Buchanan, G. Rapid Document Navigation for Information Triage Support. In *JCDL '07: Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*, Vancouver, BC, Canada, 2007. ACM, pages 503–503. ISBN: 978-1-59593-644-8. doi: 10.1145/1255175.1255303.
- [32] Buchanan, G and Loizides, F. *Research and Advanced Technology for Digital Libraries*, volume 4675/2007 of *Lecture Notes in Computer Science*, chapter Investigating Document Triage on Paper and Electronic Media, pages 416–427. Springer Berlin/Heidelberg, August 2007. ISBN: 978-3-540-74850-2. doi: 10.1007/978-3-540-74851-9.
- [33] Buchanan, G and Owen, T. Improving Navigation Interaction in Digital Documents. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, Pittsburgh PA, PA, USA, 2008. ACM, pages 389–392. ISBN: 978-1-59593-998-2. doi: 10.1145/1378889.1378959.
- [34] Buxton, W and Myers, B. A Study in Two-Handed Input. In *CHI '86: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Boston, Massachusetts, United States, 1986. ACM, pages 321–326. ISBN: 0-89791-180-6. doi: 10.1145/22627.22390.
- [35] Buxton, W. A. S. Two-Handed Document Navigation. *XEROX Disclosure Journal*, 19(2):103–108, March/April 1994.

- [36] Byrd, D. A Scrollbar-Based Visualization for Document Navigation. In *DL '99: Proceedings of the Fourth ACM Conference on Digital Libraries*, Berkeley, California, United States, 1999. ACM, pages 122–129. ISBN: 1-58113-145-3. doi: 10.1145/313238.313283.
- [37] Byrne, M. D, John, B. E, Wehrle, N. S, and Crow, D. C. The Tangled Web We Wove: A Taskonomy of WWW Use. In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, United States, 1999. ACM, pages 544–551. ISBN: 0-201-48559-1. doi: 10.1145/302979.303154.
- [38] Card, S. K, English, W. K, and Burr, B. J. Evaluation of Mouse, Rate-Controlled Isometric Joystick, Step Keys and Text Keys for Text Selection on a CRT. *Ergonomics*, 21(8):601–613, August 1978.
- [39] Catledge, L. D and Pitkow, J. E. Characterizing Browsing Strategies in the World-Wide Web. In *Proceedings of the Third International World-Wide Web Conference on Technology, Tools and Applications*, Darmstadt, Germany, 1995. Elsevier North-Holland, Inc., pages 1065–1073. ISBN: 0-169-7552. doi: 10.1016/0169-7552(95)00043-7.
- [40] Ceaparu, I, Lazar, J, Bessiere, K, Robinson, J, and Shneiderman, B. Determining causes and severity of end-user frustration. *International Journal of Human-Computer Interaction*, 17(3):333–356, September 2004.
- [41] Cechanowicz, J, Irani, P, and Subramanian, S. Augmenting the Mouse with Pressure Sensitive Input. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007. ACM, pages 1385–1394. ISBN: 978-1-59593-593-9. doi: 10.1145/1240624.1240835.
- [42] Chen, N, Guimbretiere, F, Lewis, C, and Agrawala, M. Enhancing Document Navigation Tasks with a Dual-Display Electronic Reader. *Demo at the ACM Symposium on User Interface Software and Technology 2007*, 7–10 October 2007. Available online: <http://www.acm.org/uist/archive/adjunct/2007/pdf/demos/p37-chen.pdf>, last accessed 5 December 2008.

- [43] Chen, N, Guimbretiere, F, Dixon, M, Lewis, C, and Agrawala, M. Navigation Techniques for Dual-Display E-Book Readers. In *CHI '08: Proceedings of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 2008. ACM, pages 1779–1788. ISBN: 978-1-60558-011-1. doi: 10.1145/1357054.1357331.
- [44] Chimera, R. Value Bars: An Information Visualization and Navigation Tool for Multi-Attribute Listings. In *CHI '92: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Monterey, California, United States, 1992. ACM, pages 293–294. ISBN: 0-89791-513-5. doi: 10.1145/142750.142817.
- [45] Chipman, L. E, Bederson, B. B, and Golbeck, J. A. SlideBar: Analysis of a Linear Input Device. *Journal of Behavior and Information Technology*, 23(1):1–9, April 2003.
- [46] Churchill, E. F, Trevor, J, Bly, S, Nelson, L, and Cubranic, D. Anchored Conversations: Chatting in the Context of a Document. In *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, The Hague, The Netherlands, 2000. ACM, pages 454–461. ISBN: 1-58113-216-6. doi: 10.1145/332040.332475.
- [47] Cistron. Little Brother Homepage. Available online: <http://littlebrother.sourceforge.net/>, last accessed 23 September 2008.
- [48] Claypool, M, Le, P, Wased, M, and Brown, D. Implicit Interest Indicators. In *IUI '01: Proceedings of the 6th International Conference on Intelligent User Interfaces*, Santa Fe, New Mexico, United States, 2001. ACM, pages 33–40. ISBN: 1-58113-325-1. doi: 10.1145/359784.359836.
- [49] Cockburn, A and McKenzie, B. What Do Web Users Do? An Empirical Analysis of Web Use. *International Journal of Human-Computer Studies*, 54(6):903–922, 2001. ISSN: 1071-5819. doi: 10.1006/ijhc.2001.0459.
- [50] Cockburn, A and Savage, J. Comparing Speed-Dependent Automatic Zooming with Traditional Scroll, Pan and Zoom Methods. In *People*

and Computers XVII: Proceedings of the 2003 British Computer Society Conference on Human-Computer Interaction, Bath, UK, September 2003. pages 87–102.

- [51] Cockburn, A, Greenberg, S, McKenzie, B, JasonSmith, M, and Kaasten, S. WebView: A Graphical Aid for Revisiting Web Pages. In *OZCHI '99: Proceedings of the Australian Conference on Human Computer Interaction*, Wagga Wagga, Australia, November 28–30 1999. pages 15–22.
- [52] Cockburn, A, Savage, J, and Wallace, A. Tuning and Testing Scrolling Interfaces that Automatically Zoom. In *CHI '05: Proceeding of the SIGCHI Conference on Human Factors in Computing Systems*, Portland, Oregon, USA, 2005. ACM, pages 71–80. ISBN: 1-58113-998-5. doi: 10.1145/1054972.1054983.
- [53] Cockburn, A, Gutwin, C, and Alexander, J. Faster Document Navigation with Space-Filling Thumbnails. In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006. ACM, pages 1–10. ISBN: 1-59593-372-7. doi: 10.1145/1124772.1124774.
- [54] Cockburn, A, Gutwin, C, and Greenberg, S. A Predictive Model of Menu Performance. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007. ACM, pages 627–636. ISBN: 978-1-59593-593-9. doi: 10.1145/1240624.1240723.
- [55] Cockburn, A, Karlson, A, and Bederson, B. A Review of Overview+Detail, Zooming and Focus+Context Interfaces. *ACM Computing Surveys*, 41(1): 1–31, 2008.
- [56] Collins, A and Gentner, D. A Framework for a Cognitive Theory of Writing. In Gregg, L. W and Steinberg, E. R, editors, *Cognitive Processes in Writing*, pages 51–72. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1980.

- [57] Cox, D. A, Chugh, J. S, Gutwin, C, and Greenberg, S. The Usability of Transparent Overview Layers. In *CHI '98: Summary Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, California, United States, 1998. ACM, pages 301–302. ISBN: 1-58113-028-7. doi: 10.1145/286498.286777.
- [58] CprinGold Software. CprinGold Software Homepage. Available online: <http://www.cpringold.com>, last accessed 6 October 2008.
- [59] Dearman, D, MacKay, B, Inkpen, K, and Watters, C. Touch-n-Go: Supporting Screen Navigation on Handheld Computers. Technical report, Faculty of Computer Science, Dalhousie University, July 2005. Available online: <http://cs.dal.ca/research/techreports/2005/CS-2005-08.pdf>, last accessed 5 December 2008.
- [60] DeLine, R, Czerwinski, M, Meyers, B, Venolia, G, Drucker, S, and Robertson, G. Code Thumbnails: Using Spatial Memory to Navigate Source Code. In *VLHCC '06: Symposium on Visual Languages and Human-Centric Computing 2006*, Brighton, UK, 2006. IEEE Computer Society, pages 11–18. ISBN: 0-7695-2586-5. doi: 10.1109/VLHCC.2006.14.
- [61] Dieberger, A and Russell, D. M. Exploratory Navigation in Large Multimedia Documents using Context Lenses. In *HICSS '02: Proceedings of the 35th Annual Hawaii International Conference on System Sciences, 2002*, volume 4. IEEE Computer Society, January 2002, pages 911–917. ISBN: 0-7695-1435-9. doi: 10.1109/HICSS.2002.994058.
- [62] Dillon, A. Reading From Paper versus Screens: A Critical Review of the Empirical Literature. *Ergonomics*, 35(10):1297–1326, October 1992. doi: 10.1080/00140139208967394.
- [63] Dillon, A, Kleinman, L, Choi, G. O, and Bias, R. Visual Search and Reading Tasks using ClearType and Regular Displays: Two Experiments. In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006. ACM, pages 503–511. ISBN: 1-59593-372-7. doi: 10.1145/1124772.1124849.

- [64] Dix, A. Hands Across the Screen. In *Interfaces*, volume 37, pages 19–22. Spring 1998.
- [65] Donskoy, M and Kaptelinin, V. Window Navigation With and Without Animation: A Comparison of Scroll Bars, Zoom, and Fisheye View. In *CHI '97: Extended Abstracts on Human Factors in Computing Systems*, Atlanta, Georgia, 1997. ACM, pages 279–280. ISBN: 0-89791-926-2. doi: 10.1145/1120212.1120393.
- [66] Douglas, S. A, Kirkpatrick, A. E, and MacKenzie, I. S. Testing Pointing Device Performance and User Assessment with the ISO 9241, Part 9 Standard. In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, United States, 1999. ACM, pages 215–222. ISBN: 0-201-48559-1. doi: 10.1145/302979.303042.
- [67] Eibl, J. KDiff3 Homepage. Available online: <http://kdiff3.sourceforge.net/>, last accessed 9 October 2008.
- [68] Ellis, R. D, Jankowski, T. B, Jasper, J. E, and Tharuvai, B. S. Listener: A Tool for Client-Side Investigation fo Hypermedia Navigation Behavior. *Behavior Research Methods, Instruments, & Computers*, 30(4):573–582, 1998.
- [69] English, W. K, Engelbart, D. C, and Berman, M. K. Display-Selection Techniques for Text Manipulation. *IEEE Transactions on Human Factors in Electronics*, HFE-8(1):5–15, 1967. ISSN: 0096-249X.
- [70] Eslambolchilar, P and Murray-Smith, R. Tilt-Based Automatic Zooming and Scaling in Mobile Devices: A State-Space Implementation. In Brewster, S and Dunlop, M, editors, *Proceedings of Mobile HCI 2004*, volume 3160/2004. Springer Berlin / Heidelberg, 2004, pages 120–131.
- [71] Eslambolchilar, P, Williamson, J, and Murray-Smith, R. Multimodal Feedback for Tilt Controlled Speed Dependent Automatic Zooming. In *Pro-*

ceedings of the 17th Annual ACM Symposium on User Interface Software and Technology. ACM, 2004, pages 34–35.

- [72] Fällmana, D, Lund, A, and Wiberg, M. ScrollPad: Tangible Scrolling with Mobile Devices. In *HICSS '04: Proceedings of the 37th Annual Hawaii International Conference on System Sciences - Track 9*, 5–8 January 2004. ISBN: 0-7695-2056-1.
- [73] Findlater, L and McGrenere, J. A Comparison of Static, Adaptive, and Adaptable Menus. In *CHI '04: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Vienna, Austria, 2004. ACM, pages 89–96. ISBN: 1-58113-702-8. doi: 10.1145/985692.985704.
- [74] Fitts, P. M. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.
- [75] Flower, L and Hayes, J. R. A Cognitive Process Theory of Writing. *College Composition and Communication*, 32(4):365–387, Dec 1981.
- [76] Foo Labs. Xpdf Homepage. Available online: <http://www.foolabs.com/xpdf/>, last accessed 23 October 2008.
- [77] Food and Agriculture Organization of the United Nations. *The Community's Toolbox: The Idea, Methods and Tools for Participatory Assessment, Monitoring and Evaluation in Community Forestry*, 1990. Available online: <http://www.fao.org/docrep/x5307e/x5307e08.htm>, last accessed 7 October 2008.
- [78] Free Software Foundation. GNU Emacs Manual. Available online: http://www.gnu.org/software/emacs/manual/html_mono/emacs.html, last accessed 15 November 2008.
- [79] Furnas, G. W. Generalized Fisheye Views. *SIGCHI Bulletin*, 17(4):16–23, 1986. ISSN: 0736-6906. doi: 10.1145/22339.22342.

- [80] Ginsburg, A, Marks, J, and Shieber, S. A Viewer for PostScript Documents. In *UIST '96: Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*, Seattle, Washington, United States, 1996. ACM, pages 31–32. ISBN: 0-89791-798-7. doi: 10.1145/237091.237095.
- [81] Google. Google Maps. Available online: <http://maps.google.com>, last accessed 23 October 2008.
- [82] Google. Google Desktop – Features. Available online: <http://desktop.google.com/features.html>, last accessed 26 October 2008.
- [83] Google. Google Chrome – Download a New Browser. Available online: <http://www.google.com/chrome/>, last accessed 27 October 2008.
- [84] Google. Google. Available online: <http://www.google.com>, last accessed 28 October 2008.
- [85] Google. Google Docs. Available online: <http://docs.google.com>, last accessed 11 May 2009.
- [86] Graham, J. The Reader’s Helper: A Personalized Document Reading Environment. In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, United States, 1999. ACM, pages 481–488. ISBN: 0-201-48559-1. doi: 10.1145/302979.303139.
- [87] Greenberg, S and Witten, I. H. Supporting Command Reuse: Empirical Foundations and Principles. *International Journal of Man Machine Studies*, 39(3):353–390, September 1993.
- [88] Grossman, T, Dragicevic, P, and Balakrishnan, R. Strategies for Accelerating On-line Learning of Hotkeys. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007. ACM, pages 1591–1600. ISBN: 978-1-59593-593-9. doi: 10.1145/1240624.1240865.

- [89] Guiard, Y, Beaudouin-Lafon, M, and Mottet, D. Navigation as Multiscale Pointing: Extending Fitts' Model to Very High Precision Tasks. In *CHI '99: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, United States, 1999. ACM, pages 450–457. ISBN: 0-201-48559-1. doi: 10.1145/302979.303128.
- [90] Guiard, Y, Beaudouin-Lafon, M, Bastin, J, Pasveer, D, and Zhai, S. View Size and Pointing Difficulty in Multi-Scale Navigation. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 2004. ACM, pages 117–124. ISBN: 1-58113-867-9. doi: 10.1145/989863.989881.
- [91] Guiard, Y, Beaudouin-Lafon, M, Du, Y, Appert, C, Fekete, J.-D, and Chapuis, O. Shakespeare's Complete Works as a Benchmark for Evaluating Multiscale Document Navigation Techniques. In *BELIV '06: Proceedings of the 2006 AVI Workshop on Beyond Time and Errors*, Venice, Italy, 2006. ACM, pages 1–6. ISBN: 1-59593-562-2. doi: 10.1145/1168149.1168165.
- [92] Guiard, Y, Chapuis, O, Du, Y, and Beaudouin-Lafon, M. Allowing Camera Tilts for Document Navigation in the Standard GUI: A Discussion and an Experiment. In *AVI '06: Proceedings of the Working Conference on Advanced Visual Interfaces*, Venezia, Italy, 2006. ACM, pages 241–244. ISBN: 1-59593-353-0. doi: 10.1145/1133265.1133312.
- [93] Guiard, Y, Du, Y, and Chapuis, O. Quantifying Degree of Goal Directness in Document Navigation: Application to the Evaluation of the Perspective-Drag Technique. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007. ACM, pages 327–336. ISBN: 978-1-59593-593-9. doi: 10.1145/1240624.1240679.
- [94] Gutwin, C, Greenberg, S, and Roseman, M. Workspace Awareness Support with Radar Views. In *CHI '96: Conference Companion on Human Factors in Computing Systems*, Vancouver, British Columbia, Canada, 1996. ACM, pages 210–211. ISBN: 0-89791-832-0. doi: 10.1145/257089.257286.

- [95] Hachet, M, Pouderoux, J, Guitton, P, and Gonzato, J.-C. TangiMap: A Tangible Interface for Visualization of Large Documents on Handheld Computers. In *GI '05: Proceedings of Graphics Interface 2005*, Victoria, British Columbia, 2005. Canadian Human-Computer Communications Society, pages 9–15. ISBN: 1-56881-265-5.
- [96] Harrison, B. L, Fishkin, K. P, Gujar, A, Mochon, C, and Want, R. Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces. In *CHI '98: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, California, United States, 1998. ACM Press/Addison-Wesley Publishing Co., pages 17–24. ISBN: 0-201-30987-4. doi: 10.1145/274644.274647.
- [97] Hart, S and Staveland, L. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In Hancock, P and Meshkati, N, editors, *Human Mental Workload*, pages 139–183. Elsevier Science, 1988.
- [98] Hawkey, K and Inkpen, K. Web Browsing Today: The Impact of Changing Contexts on User Activity. In *CHI '05: Extended Abstracts on Human Factors in Computing Systems*, Portland, OR, USA, 2005. ACM, pages 1443–1446. ISBN: 1-59593-002-7. doi: 10.1145/1056808.1056937.
- [99] Hill, W. C, Hollan, J. D, Wroblewski, D, and McCandless, T. Edit Wear and Read Wear. In *CHI '92: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Monterey, California, United States, 1992. ACM, pages 3–9. ISBN: 0-89791-513-5. doi: 10.1145/142750.142751.
- [100] Hinckley, K, Cutrell, E, Bathiche, S, and Muss, T. Quantitative Analysis of Scrolling Techniques. In *CHI '02: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Minneapolis, Minnesota, USA, 2002. ACM, pages 65–72. ISBN: 1-58113-453-3. doi: 10.1145/503376.503389.
- [101] Hoeben, A and Stappers, P. J. Flicking Through Page-Based Documents with Thumbnail Sliders and Electronic Dog-Ears. In *CHI '00: Extended*

- Abstracts on Human Factors in Computing Systems*, The Hague, The Netherlands, 2000. ACM, pages 191–192. ISBN: 1-58113-248-4. doi: 10.1145/633292.633397.
- [102] Hong, J. I, Heer, J, Waterson, S, and Landay, J. A. WebQuilt: A Proxy-Based Approach to Remote Web Usability Testing. *ACM Transactions on Information Systems*, 19(3):263–285, 2001. ISSN: 1046-8188. doi: 10.1145/502115.502118.
- [103] Hornbæk, K and Frøkjær, E. Reading of Electronic Documents: the Usability of Linear, Fisheye, and Overview+Detail Interfaces. In *CHI '01: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Seattle, Washington, United States, 2001. ACM, pages 293–300. ISBN: 1-58113-327-8. doi: 10.1145/365024.365118.
- [104] Hunt, J. W and McIlroy, M. D. An Algorithm for Differential File Comparison. Bell Telephone Laboratories, Computer Science Technical Report #41, 1976.
- [105] Hunter, W. J and Begoray, J. A Framework for the Activities Involved in the Writing Process. *The Writing Notebook*, 8(1), 1990.
- [106] Hutchings, D. R, Smith, G, Meyers, B, Czerwinski, M, and Robertson, G. Display Space Usage and Window Management Operation Comparisons Between Single Monitor and Multiple Monitor Users. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 2004. ACM, pages 32–39. ISBN: 1-58113-867-9. doi: 10.1145/989863.989867.
- [107] Hutchins, E. L, Hollan, J. D, and Norman, D. A. Direct manipulation interfaces. *Human-Computer Interaction*, 1(4):311–338, February 1985.
- [108] IBM Corporation. ScrollPoint Pro Mouse – Overview. Available online: <http://www-307.ibm.com/pc/support/site.wss/MIGR-4GUTHP.html>, last accessed 8 October 2008.

- [109] Igarashi, T and Hinckley, K. Speed-Dependent Automatic Zooming for Browsing Large Documents. In *UIST '00: Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, San Diego, California, United States, 2000. ACM, pages 139–148. ISBN: 1-58113-212-3. doi: 10.1145/354401.354435.
- [110] Iolo Technologies, LLC. Macro Magic. Available online: <http://www.iolo.com/mm/>, last accessed 6 October 2008.
- [111] Ishak, E. W and Feiner, S. K. Content-Aware Scrolling. In *UIST '06: Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology*, Montreux, Switzerland, 2006. ACM, pages 155–158. ISBN: 1-59593-313-1. doi: 10.1145/1166253.1166277.
- [112] Jakobsen, M. R and Hornbæk, K. Evaluating a Fisheye View of Source Code. In *CHI '06: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006. ACM, pages 377–386. ISBN: 1-59593-372-7. doi: 10.1145/1124772.1124830.
- [113] Jitbit Software. Jitbit Software Homepage. Available online: <http://www.jitbit.com>, last accessed 6 October 2008.
- [114] Johnson, W, Jellinek, H, Leigh Klotz, J, Rao, R, and Card, S. K. Bridging the Paper and Electronic Worlds: The Paper User Interface. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993. ACM, pages 507–512. ISBN: 0-89791-575-5. doi: 10.1145/169059.169445.
- [115] Joyce, J. *Ulysses*. Project Gutenberg, 2008. Available online: <http://www.gutenberg.org/etext/4300>, last accessed 21 April 2009.
- [116] Joyce, J. *Dubliners*. Project Gutenberg, 2008. Available online: <http://www.gutenberg.org/etext/2814>, last accessed 22 April 2009.
- [117] Juran, J. *Quality Control Handbook*. McGraw-Hill, New York, 1951.

- [118] Kaptelinin, V, Mäntylä, T, and ström, J. A. Transient Visual Cues for Scrolling: An Empirical Study. In *CHI '02: Extended Abstracts on Human Factors in Computing Systems*, Minneapolis, Minnesota, USA, 2002. ACM, pages 620–621. ISBN: 1-58113-454-1. doi: 10.1145/506443.506513.
- [119] KeeLog. The KeeLogger: Hardware Keylogger Solutions. Available online: <http://www.keyghost.com>, last accessed 23 September 2008.
- [120] Kellar, M, Watters, C, and Shepherd, M. The Impact of Task on the Usage of Web Browser Navigation Mechanisms. In *GI '06: Proceedings of Graphics Interface 2006*, Quebec, Canada, 2006. Canadian Information Processing Society, pages 235–242. ISBN: 1-56881-308-2.
- [121] Kellar, M, Hawkey, K, Inkpen, K. M, and Watters, C. Challenges of Capturing Natural Web-based User Behaviours. *International Journal of Human-Computer Interaction*, 24(4):385–409, May 2008.
- [122] Kelly, D and Belkin, N. J. Reading Time, Scrolling and Interaction: Exploring Implicit Sources of User Preferences for Relevance Feedback. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, Louisiana, United States, 2001. ACM, pages 408–409. ISBN: 1-58113-331-6. doi: 10.1145/383952.384045.
- [123] Kelly, D and Belkin, N. J. Display Time as Implicit Feedback: Understanding Task Effects. In *SIGIR '04: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Sheffield, United Kingdom, 2004. ACM, pages 377–384. ISBN: 1-58113-881-4. doi: 10.1145/1008992.1009057.
- [124] KeyGhost Ltd. KeyGhost Keylogger. Available online: <http://www.keyghost.com>, last accessed 23 September 2008.
- [125] Kim, K.-S and Allen, B. Cognitive and Task Influences on Web Searching Behavior. *Journal of the American Society for Information Science and*

- Technology*, 53(2):109–119, 2002. ISSN: 1532-2882. doi: 10.1002/asi.10014.
- [126] Kim, S, Kim, H, Lee, B, Nam, T.-J, and Lee, W. Inflatable Mouse: Volume-Adjustable Mouse with Air-Pressure-Sensitive Input and Haptic Feedback. In *CHI '08: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Florence, Italy, 2008. ACM, pages 211–224. ISBN: 978-1-60558-011-1. doi: 10.1145/1357054.1357090.
 - [127] Klein, C and Bederson, B. B. Benefits of Animated Scrolling. In *CHI '05: Extended Abstracts on Human Factors in Computing Systems*, Portland, OR, USA, 2005. ACM, pages 1965–1968. ISBN: 1-59593-002-7. doi: 10.1145/1056808.1057068.
 - [128] Kumar, M and Winograd, T. Gaze-Enhanced Scrolling Techniques. In *UIST '07: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, Newport, Rhode Island, USA, 2007. ACM, pages 213–216. ISBN: 978-1-59593-679-2. doi: 10.1145/1294211.1294249.
 - [129] Laakso, S. A, Laakso, K.-P, and Saura, A. J. Improved Scroll Bars. In *CHI '00: Extended Abstracts on Human Factors in Computing Systems*, The Hague, The Netherlands, 2000. ACM, pages 97–98. ISBN: 1-58113-248-4. doi: 10.1145/633292.633350.
 - [130] Landauer, T. K and Nachbar, D. W. Selection from Alphabetic and Numeric Menu Trees Using a Touch Screen: Breadth, Depth, and Width. In *CHI '85: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Francisco, California, United States, 1985. ACM, pages 73–78. ISBN: 0-89791-149-0. doi: 10.1145/317456.317470.
 - [131] Lawton, D. T and Feigin, E. J. Streaming Thumbnails: Combining Low Resolution Navigation and RSVP Displays. In *CHI '00: Extended Abstracts on Human Factors in Computing Systems*, The Hague, The Netherlands, 2000. ACM, pages 159–160. ISBN: 1-58113-248-4. doi: 10.1145/633292.633381.

- [132] Lewis, C and Rieman, J. *Task-Centered User Interface Design: A Practical Introduction*. Shareware Book, 1994. Available online: <http://hcibib.org/tcuid/index.html>, last accessed 6 October 2008.
- [133] Liesaputra, V and Witten, I. H. Seeking Information in Realistic Books: A User Study. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, Pittsburgh PA, PA, USA, 2008. ACM, pages 29–38. ISBN: 978-1-59593-998-2. doi: 10.1145/1378889.1378896.
- [134] Linton, F, Joy, D, Schaefer, H.-P, and Charron, A. OWL: A Recommender System for Organization-Wide Learning. *Educational Technology & Society*, 3(1), 2000.
- [135] Logitech. TrackMan Wheel. Available online: http://www.logitech.com/index.cfm/mice_pointers/trackballs/devices/4787, last accessed 8 October 2008.
- [136] Logitech Innovation Brief. MicroGear™ Precision Scroll Wheel and SmartShift™ Technology. Available online: http://www.logitech.com/lang/pdf/ib-microgear_and_smartshift_EN.pdf, last accessed 15 November 2008.
- [137] Loizides, F and Buchanan, G. R. The Myth of Find: User Behaviour and Attitudes Towards the Basic Search Feature. In *JCDL '08: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, Pittsburgh PA, PA, USA, 2008. ACM, pages 48–51. ISBN: 978-1-59593-998-2. doi: 10.1145/1378889.1378898.
- [138] Lunzer, E. *The Effective Use of Reading*, chapter From Learning to Read to Reading to Learn. Heinemann Educational Books Ltd, London, 1979.
- [139] MacArthur, C. A, Graham, S, and Fitzgerald, J. *Handbook of Writing Research*, chapter Introduction, pages 1–7. The Guilford Press, 2008.
- [140] Mackay, W. E, Appert, C, Beaudouin-Lafon, M, Chapuis, O, Du, Y, Fekete, J.-D, and Guiard, Y. Touchstone: Exploratory Design of Experiments. In

- CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007. ACM, pages 1425–1434. ISBN: 978-1-59593-593-9. doi: 10.1145/1240624.1240840.
- [141] MacKenzie, I. S. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Human-Computer Interaction*, 7(1):91–139, March 1992. doi: 10.1207/s15327051hci0701_3.
- [142] MacKenzie, I. S, Sellen, A, and Buxton, W. A. S. A Comparison of Input Devices in Element Pointing and Dragging Tasks. In *CHI '91: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, New Orleans, Louisiana, United States, 1991. ACM, pages 161–166. ISBN: 0-89791-383-3. doi: 10.1145/108844.108868.
- [143] Marsh, K. Win32 Hooks. Available online: <http://msdn.microsoft.com/en-us/library/ms997537.aspx>, last accessed 30 October 2008.
- [144] Marshall, C. C and Bly, S. Turning the Page on Navigation. In *JCDL '05: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, Denver, CO, USA, 2005. ACM, pages 225–234. ISBN: 1-58113-876-8. doi: 10.1145/1065385.1065438.
- [145] Masoodian, M, McKoy, S, Rogers, B, and Ware, D. DeepDocument: Use of a Multi-Layered Display to Provide Context Awareness in Text Editing. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 2004. ACM, pages 235–239. ISBN: 1-58113-867-9. doi: 10.1145/989863.989902.
- [146] Masui, T, Kashiwagi, K, and IV, G. R. B. Elastic Graphical Interfaces for Precise Data Manipulation. In *CHI '95: Conference Companion on Human Factors in Computing Systems*, Denver, Colorado, United States, 1995. ACM, pages 143–144. ISBN: 0-89791-755-3. doi: 10.1145/223355.223471.
- [147] Mayo, E. *The Human Problems of an Industrial Civilization*. Cambridge, MA: Harvard University Press, 1933.

- [148] McCrickard, D. S and Catrambone, R. Beyond the Scrollbar: An Evolution and Evaluation of Alternative Navigation Techniques. In *Proceedings of the IEEE Symposium on Visual Languages (VL '99)*, Tokyo, Japan, 1999. pages 270–277.
- [149] McGrenere, J. *The Design and Evaluation of Multiple Interfaces: A Solution for Complex Software*. PhD thesis, University of Toronto, 2002.
- [150] Melchior, M. Perceptually Guided Scrolling for Reading Continuous Text on Small Screen Devices. In Dunlop, M. D and Brewster, S. A, editors, *Proceedings of Mobile HCI 2001: Third International Workshop on Human Computer Interaction with Mobile Devices*, September 2001. Available online: http://personal.cis.strath.ac.uk/~mdd/mobilehci01/procs/melchior_cr.pdf, last accessed 5 December 2008.
- [151] Microsoft. Live Search. Available online: <http://www.live.com>, last accessed 28 October 2008.
- [152] Microsoft Corporation. Microsoft Office Word Homepage, 2008. Available online: <http://www.microsoft.com/word/>, last accessed 6 October 2008.
- [153] Microsoft Corporation. Microsoft Office Online Homepage, 2008. Available online: <http://www.microsoft.com/office/>, last accessed 6 October 2008.
- [154] Microsoft Corporation. Visual Studio Development System, 2008. Available online: <http://www.microsoft.com/vstudio/>, last accessed 18 July 2008.
- [155] Microsoft Corporation. Windows XP Home Page, 2008. Available online: <http://www.microsoft.com/windows/windows-xp/>, last accessed 18 July 2008.
- [156] Microsoft Corporation. Internet Explorer Homepage, 2008. Available online: <http://www.microsoft.com/ie/>, last accessed 6 October 2008.

- [157] Microsoft Corporation. Microsoft Customer Experience Improvement Program, 2008. Available online: <http://www.microsoft.com/products/ceip/>, last accessed 6 October 2008.
- [158] Microsoft Corporation. What is ClearType?, 2002. Available online: <http://www.microsoft.com/typography/WhatIsClearType.aspx>, last accessed 12 September 2008.
- [159] Microsoft Corporation. Overview: WinDiff, 2008. Available online: [http://msdn.microsoft.com/en-us/library/aa242739\(VS.60\).aspx](http://msdn.microsoft.com/en-us/library/aa242739(VS.60).aspx), last accessed 9 October 2008.
- [160] Microsoft Corporation. Microsoft Office Excel, 2008. Available online: <http://www.microsoft.com/excel/>, last accessed 10 October 2008.
- [161] Microsoft Corporation. Explore the Features: Search and Organization, 2008. Available online: <http://www.microsoft.com/windows/windows-vista/features/search-and-organization.aspx>, last accessed 26 October 2008.
- [162] Microsoft Corporation. Home Page: Spy++, 2008. Available online: [http://msdn.microsoft.com/en-us/library/aa264396\(VS.60\).aspx](http://msdn.microsoft.com/en-us/library/aa264396(VS.60).aspx), last accessed 29 October 2008.
- [163] Microsoft Corporation. Active Accessibility 2.0 SDK Tools, 2008. Available online: <http://www.microsoft.com/downloads/details.aspx?familyid=3755582a-a707-460a-bf21-1373316e13f0&displaylang=en>, last accessed 29 October 2008.
- [164] Microsoft Corporation. About Messages and Message Queues, 2008. Available online: <http://msdn.microsoft.com/en-us/library/ms644927.aspx>, last accessed 29 October 2008.
- [165] Microsoft Corporation. Virtual-Key Codes, 2008. Available online: <http://msdn.microsoft.com/en-us/library/ms645540.aspx>, last accessed 30 October 2008.

- [166] Microsoft Corporation. Microsoft Active Accessibility, 2008. Available online: <http://msdn2.microsoft.com/en-us/library/ms697707.aspx>, last accessed 30 October 2008.
- [167] Microsoft Corporation. Scroll Bar Controls in Win32, 2008. Available online: <http://msdn.microsoft.com/en-us/library/ms997557.aspx>, last accessed 3 December 2008.
- [168] Microsoft Corporation. About Windows, 2008. Available online: [http://msdn.microsoft.com/en-us/library/ms632597\(VS.85\).aspx#window_handle](http://msdn.microsoft.com/en-us/library/ms632597(VS.85).aspx#window_handle), last accessed 17 December 2008.
- [169] Microsoft Corporation. Windows Vista Home Page, 2009. Available online: <http://www.microsoft.com/windows/windows-vista/>, last accessed 13 May 2009.
- [170] Microsoft Corporation. .NET Framework Developer Documentation: Track Class, 2009. Available online: <http://msdn2.microsoft.com/en-us/library/system.windows.controls.primitives.track.aspx>.
- [171] Mooser, J, You, S, and Neumann, U. Large Document, Small Screen: A Camera Driven Scroll and Zoom Control for Mobile Devices. In *SI3D '08: Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, Redwood City, California, 2008. ACM, pages 27–34. ISBN: 978-1-59593-983-8. doi: 10.1145/1342250.1342254.
- [172] Moscovich, T and Hughes, J. F. Navigating Documents with the Virtual Scroll Ring. In *UIST '04: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, Santa Fe, NM, USA, 2004. ACM, pages 57–60. ISBN: 1-58113-957-8. doi: 10.1145/1029632.1029642.
- [173] Mozilla. Firefox Web Browser Homepage. Available online: <http://www.mozilla.com/firefox/>, last accessed 23 October 2008.

- [174] Myers, B. A, Lie, K. P, and Yang, B.-C. Two-handed input using a pda and a mouse. In *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, The Hague, The Netherlands, 2000. ACM, pages 41–48. ISBN: 1-58113-216-6. doi: 10.1145/332040.332405.
- [175] National Center for Supercomputing Applications. About NCSA Mosaic. Available online: <http://www.ncsa.uiuc.edu/Projects/mosaic.html>, last accessed 6 October 2008.
- [176] NCH Swift Sound. Express Scribe Transcription Playback Software Homepage. Available online: <http://www.nch.com.au/scribe/>, last accessed 23 September 2008.
- [177] NetBeans. NetBeans IDE. Available online: <http://www.netbeans.org/>, last accessed 8 October 2008.
- [178] Oakley, I, Ängeslevä, J, Hughes, S, and O'Modhrain, S. Tilt and Feel: Scrolling with Vibrotactile Display. In *Proceedings of EuroHaptics 2004*, June 5–7 2004, pages 316–323.
- [179] Obendorf, H, Weinreich, H, Herder, E, and Mayer, M. Web Page Revisitation Revisited: Implications of a Long-Term Click-Stream Study of Browser Usage. In *CHI '07: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007. ACM, pages 597–606. ISBN: 978-1-59593-593-9. doi: 10.1145/1240624.1240719.
- [180] O'Hara, K. Towards a Typology of Reading Goals. Technical Report EPC-1996-107, Rank Xerox Research Centre, 1996.
- [181] O'Hara, K and Sellen, A. A Comparison of Reading Paper and On-Line Documents. In *CHI '97: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Atlanta, Georgia, United States, 1997. ACM, pages 335–342. ISBN: 0-89791-802-9. doi: 10.1145/258549.258787.

- [182] O'Hara, K, Smith, F, Newman, W, and Sellen, A. Student Readers' use of Library Documents: Implications for Library Technologies. In *CHI '98: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Los Angeles, California, United States, 1998. ACM Press/Addison-Wesley Publishing Co., pages 233–240. ISBN: 0-201-30987-4. doi: 10.1145/274644.274678.
- [183] OpenOffice.org. Calc. Available online: <http://www.openoffice.org/product/calc.html>, last accessed 10 October 2008.
- [184] Öquist, G and Lundin, K. Eye Movement Study of Reading Text on a Mobile Phone using Paging, Scrolling, Leading, and RSVP. In *MUM '07: Proceedings of the 6th International Conference on Mobile and Ubiquitous Multimedia*, Oulu, Finland, 2007. ACM, pages 176–183. ISBN: 978-1-59593-916-6. doi: 10.1145/1329469.1329493.
- [185] Pearsall, J, editor. *The Concise Oxford Dictionary*. Oxford University Press, 10th edition, 1999.
- [186] Perlin, K and Fox, D. Pad: An Alternative Approach to the Computer Interface. In *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1993, pages 57–64. ISBN: 0-89791-601-8. doi: 10.1145/166117.166125.
- [187] Python Software Foundation. Python Programming Language – Official Website. Available online: <http://www.python.org>, last accessed 18 July 2008.
- [188] Ramos, G and Balakrishnan, R. Zliding: Fluid Zooming and Sliding for High Precision Parameter Manipulation. In *UIST '05: Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, Seattle, WA, USA, 2005. ACM, pages 143–152. ISBN: 1-59593-271-2. doi: 10.1145/1095034.1095059.
- [189] Reeder, R. W, Pirolli, P, and Card, S. K. WebLogger: A Data Collection Tool for Web-use Studies. Technical report,

- Xerox Palo Alto Research Center, 2000. Available online: <http://www2.parc.com/istl/projects/uir/publications/items/UIR-2000-06-Reeder-TechReport-WebLogger.pdf>, last accessed 5 December 2008.
- [190] Ren, X, Zhou, X, and Liu, Z. An Empirical Evaluation of Seven Mice for Scrolling Tasks. *ICMA 2007: International Conference on Mechatronics and Automation*, pages 582–586, August 2007. doi: 10.1109/ICMA.2007.4303608.
- [191] ResearchWare Inc. HyperTRANSCRIBE. Available online: <http://www.researchware.com/ht/index.html>, last accessed 23 September 2008.
- [192] Rieman, J. The Diary Study: A Workplace-Oriented Research Tool to Guide Laboratory Efforts. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993. ACM, pages 321–326. ISBN: 0-89791-575-5. doi: 10.1145/169059.169255.
- [193] Robertson, G. G and Mackinlay, J. D. The Document Lens. In *UIST '93: Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, Atlanta, Georgia, United States, 1993. ACM, pages 101–108. ISBN: 0-89791-628-X. doi: 10.1145/168642.168652.
- [194] Rohman, D. G. Pre-Writing the Stage of Discovery in the Writing Process. *College Composition and Communication*, 16(2):106–112, May 1965.
- [195] Rosner, D. K, Oehlberg, L, and Ryokai, K. Studying paper use to inform the design of personal and portable technology. In *CHI '08: Extended Abstracts on Human Factors in Computing Systems*, Florence, Italy, 2008. ACM, pages 3405–3410. ISBN: 978-1-60558-012-X. doi: 10.1145/1358628.1358865.
- [196] Savage, J and Cockburn, A. Comparing Automatic and Manual Zooming Methods for Acquiring Off-Screen Targets. In *People and Computers XIX*:

- Proceedings of the 2005 British Computer Society Conference on Human-Computer Interaction*, Edinburgh, UK, 2005. Springer London, pages 439–454. ISBN: 978-1-84628-192-1. doi: 10.1007/1-84628-249-7_28.
- [197] Schenk, C. MiKTeX Project Page. Available online: <http://www.miktex.org>, last accessed 23 October 2008.
- [198] Sears, A and Shneiderman, B. Split Menus: Effectively Using Selection Frequency to Organize Menus. *ACM Transactions on Computer-Human Interaction*, 1(1):27–51, 1994. ISSN: 1073-0516. doi: 10.1145/174630.174632.
- [199] Shneiderman, B. *Designing the User Interface. Strategies for Effective Human-Computer Interaction*. Addison Wesley, second edition, 1992.
- [200] Siio, I. Scroll Display: Pointing Device for Palmtop Computers. In *Proceedings of the 3rd Asia Pacific Conference on Computer Human Interaction*, 15–17 July 1998, pages 243–248. ISBN: 0-8186-8347-3. doi: 10.1109/APCHI.1998.704324.
- [201] Simon, H. Satisficing. In Eatwell, J, Millgate, M, and Newman, P, editors, *The New Palgrave: A Dictionary of Economics*, pages 243–245. Stockton Press, New York, 1987.
- [202] Skopik, A and Gutwin, C. Improving Revisitation in Fisheye Views with Visit Wear. In *CHI '05: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Portland, Oregon, USA, 2005. ACM, pages 771–780. ISBN: 1-58113-998-5. doi: 10.1145/1054972.1055079.
- [203] Smith, G. M and Schraefel, M. C. The Radial Scroll Tool: Scrolling Support for Stylus- or Touch-Based Document Navigation. In *UIST '04: Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, Santa Fe, NM, USA, 2004. ACM, pages 53–56. ISBN: 1-58113-957-8. doi: 10.1145/1029632.1029641.

- [204] Smith, R. B and Taivalsaari, A. Generalized and Stationary Scrolling. In *UIST '99: Proceedings of the 12th Annual ACM Symposium on User Interface Software and Technology*, Asheville, North Carolina, United States, 1999. ACM, pages 1–9. ISBN: 1-58113-075-9. doi: 10.1145/320719.322577.
- [205] Song, H, Qi, Y, Xiao, L, Zhu, T, and Curran, E. P. *Human Interface and the Management of Information, Methods, Techniques and Tools in Information Design*, volume 4557/2007 of *Lecture Notes in Computer Science*, chapter ActiveScrollbar: A Scroll Bar With Direct Scale Ratio Control, pages 354–358. Springer Berlin/Heidelberg, August 2007. ISBN: 978-3-540-73344-7. doi: 10.1007/978-3-540-73345-4.
- [206] Sony Electronics Inc. Reader Digital Book. Available online: <http://www.sonymstyle.com/reader/>, last accessed 20 November 2008.
- [207] Spence, B, Witkowski, M, Fawcett, C, Craft, B, and de Bruijn, O. Image Presentation in Space and Time: Errors, Preferences and Eye-Gaze Activity. In *AVI '04: Proceedings of the Working Conference on Advanced Visual Interfaces*, Gallipoli, Italy, 2004. ACM, pages 141–149.
- [208] Steimle, J, Brdiczka, O, and Mühlhäuser, M. Digital Paper Bookmarks: Collaborative Structuring, Indexing and Tagging of Paper Documents. In *CHI '08: Extended Abstracts on Human Factors in Computing Systems*, Florence, Italy, 2008. ACM, pages 2895–2900. ISBN: 978-1-60558-012-X. doi: 10.1145/1358628.1358780.
- [209] Suh, B, Woodruff, A, Rosenholtz, R, and Glass, A. Popout Prism: Adding Perceptual Principles to Overview+Detail Document Interfaces. In *CHI '02: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Minneapolis, Minnesota, USA, 2002. ACM, pages 251–258. ISBN: 1-58113-453-3. doi: 10.1145/503376.503422.
- [210] Sun, L and Guimbretière, F. Flipper: A New Method of Digital Document Navigation. In *CHI '05: Extended Abstracts on Human Factors in Com-*

- puting Systems*, Portland, OR, USA, 2005. ACM, pages 2001–2004. ISBN: 1-59593-002-7. doi: 10.1145/1056808.1057077.
- [211] Sun Microsystems, Inc. JScrollBar (Java Platform SE 6). Available online: <http://java.sun.com/javase/6/docs/api/javax/swing/JScrollBar.html>, last accessed 3 December 2008.
 - [212] Sutherland, I. E. Sketch Pad A Man-Machine Graphical Communication System. In *DAC '64: Proceedings of the SHARE Design Automation Workshop*. ACM, 1964, pages 6.329–6.346. doi: 10.1145/800265.810742.
 - [213] Tauscher, L and Greenberg, S. Revisitation Patterns in World Wide Web Navigation. In *CHI '97: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Atlanta, Georgia, United States, 1997. ACM, pages 399–406. ISBN: 0-89791-802-9. doi: 10.1145/258549.258816.
 - [214] Tauscher, L and Greenberg, S. How People Revisit Web Pages: Empirical Findings and Implications for the Design of History Systems. *International Journal of Human-Computer Studies*, 47(1):97–137, 1997. ISSN: 1071-5819. doi: 10.1006/ijhc.1997.0125.
 - [215] TechSmith Corporation. Camtasia Studio Homepage. Available online: <http://www.techsmith.com/camtasia.asp>, last accessed 6 October 2008.
 - [216] Tethys Solutions, LLC. Tethys Solutions Homepage. Available online: <http://www.tethyssolutions.com>, last accessed 6 October 2008.
 - [217] The Eclipse Foundation. Eclipse.org Homepage. Available online: <http://www.eclipse.org/>, last accessed 8 October 2008.
 - [218] The GNOME Usability Project. GNOME Human Interface Guidelines 2.0. Available online: <http://developer.gnome.org/projects/gup/hig/2.0/hig-2.0.pdf>, last accessed 22 October 2008.

- [219] Treisman, A. M and Gelade, G. A Feature-Integration Theory of Attention. *Cognitive Psychology*, 12(1):97–136, January 1980.
- [220] Trewin, S. InputLogger: General-Purpose Logging of Keyboard and Mouse Events on an Apple Macintosh. *Behavior Research Methods, Instruments and Computers*, 30(2):327–331, 1998.
- [221] Turnbull, D. WebTracker: A Tool for Understanding Web Use. Available online: <http://www.ischool.utexas.edu/~donturn/research/webtracker/>, last accessed 6 October 2008.
- [222] Ulrich, S. The Xdvi Homepage. Available online: <http://xdvi.sourceforge.net/>, last accessed 23 October 2008.
- [223] Urmila Kukreja, F. E. R William E. Stevenson. RUI: Recording User Input from Interfaces under Windows and Mac OS X. *Behavior Research Methods*, 38(4):656–659, November 2006.
- [224] Venolia, D. Facile 3D Direct Manipulation. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993. ACM, pages 31–36. ISBN: 0-89791-575-5. doi: 10.1145/169059.169065.
- [225] Weinreich, H, Obendorf, H, Herder, E, and Mayer, M. Off the Beaten Tracks: Exploring Three Aspects of Web Navigation. In *WWW '06: Proceedings of the 15th International Conference on World Wide Web*, Edinburgh, Scotland, 2006. ACM, pages 133–142. ISBN: 1-59593-323-9. doi: 10.1145/1135777.1135802.
- [226] Westerman, S, Hambly, S, Alder, C, Wyatt-Millington, C, Shryane, N, Crawshaw, C, and Hockey, G. Investigating the Human-Computer Interface Using the Datalogger. *Behavior Research Methods, Instruments and Computers*, 28(4):603–606, 1996.
- [227] Wexelblat, A and Maes, P. Footprints: History-Rich Tools for Information Foraging. In *CHI '99: Proceedings of the SIGCHI Confer-*

- ence on Human Factors in Computing Systems*, Pittsburgh, Pennsylvania, United States, 1999. ACM, pages 270–277. ISBN: 0-201-48559-1. doi: 10.1145/302979.303060.
- [228] Wherry, E. Scroll Ring Performance Evaluation. In *CHI '03: Extended Abstracts on Human Factors in Computing Systems*, Ft. Lauderdale, Florida, USA, 2003. ACM, pages 758–759. ISBN: 1-58113-637-4. doi: 10.1145/765891.765973.
- [229] William H. Gates III and Steven A. Ballmer. Shareholder Letter, Microsoft Coporation Annual Report 2008, 2008. Available online: http://www.microsoft.com/msft/reports/ar08/10k_sl_eng.html.
- [230] Wolfe, J. M. *Attention*, chapter Visual Search, pages 13–73. University College London Press, 1998.
- [231] Wolfe, J. M, Cave, K. R, and Franzel, S. L. Guided Search: An Alternative to the Feature Integration Model for Visual Search. *Journal of Experimental Psychology: Human Perception and Performance*, 15(3):419–433, 1989.
- [232] Woodruff, A, Faulring, A, Rosenholtz, R, Morrision, J, and Pirolli, P. Using Thumbnails to Search the Web. In *CHI '01: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Seattle, Washington, United States, 2001. ACM, pages 198–205. ISBN: 1-58113-327-8. doi: 10.1145/365024.365098.
- [233] Wroblewski, D. A, Hill, W. C, and McCandless, T. P. Attribute-Enhanced Scroll Bar System and Method. United States Patent 5479600, December 1995.
- [234] Wulfe, B. Managed Spy: Deliver The Power Of Spy++ To Windows Forms With Our New Tool. Available online: <http://msdn.microsoft.com/en-us/magazine/cc163617.aspx>, last accessed 29 October 2008.

- [235] Yahoo! Yahoo! Available online: <http://www.yahoo.com>, last accessed 28 October 2008.
- [236] Zhai, S, Smith, B. A, and Selker, T. Improving Browsing Performance: A Study of Four Input Devices for Scrolling and Pointing Tasks. In *INTERACT '97: Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*. Chapman & Hall, Ltd., 1997, pages 286–293. ISBN: 0-412-80950-8.
- [237] Zipf, G. K. *Human Behaviour and the Principle of Least Effort: An Introduction to Human Ecology*. Hafner Publishing Company, 1965.

Appendices

Appendix A

Information Sheet for Human Ethics Committee

THE information sheet that follows was provided to the University of Canterbury's Human Ethics Committee. It informed them of the details of the longitudinal study that monitored users' actions using the AppMonitor software.

Information Sheet for

AppMonitor: A Document Navigation Event Logger

Prepared by: Jason Alexander (Ph.D. Candidate, jma142@student.canterbury.ac.nz)
Supervisor: Assoc. Prof. Andy Cockburn (andy@cosc.canterbury.ac.nz)
Date: May 21, 2009

Purpose of Document

The purpose of this document is to give the Human Ethics Committee (HEC) a record of our intentions with regard to running software to aid our research into document navigation. Communication with the HEC Chair, Alison Loveridge (21 February 2006), confirmed that a full HEC approval would not be required, but suggested an information sheet be produced for future reference.

Purpose of Software

The AppMonitor software has been created to provide a means of gathering empirical data on document navigation. Distribution of the application to between 50 and 100 participants will allow us to collect a large amount of data over a three to six month period. This is the first stage of work on a Ph.D. thesis with the core aim of improving document navigation.

Software Operation

The AppMonitor software records navigation actions in the common Windows applications Microsoft Word and Adobe Reader. Once installed the software runs quietly in the background requiring no input from the computer user—the participant does not need to adjust their everyday computer use in any way. The list below provides examples of events recorded by the software:

- A document being opened or closed.
- Scrollbar movement: How far the scrollbar was moved and what caused the move (a click on the arrows, thumb dragged, etc).
- Items being selected from menus.
- Window actions: The resizing, moving, minimising and maximising of the application window.
- Special key presses: Page up, Page down, the arrow keys, Tab, Enter, the Function keys (F1–F12) or any key press *with a modifier* (Ctrl, Alt or the Special key held down simultaneously with a standard key).

The software does not record every keystroke entered into the monitored applications. This is important for two reasons: firstly, we do not want to unnecessarily invade user privacy and secondly, it ensures that no form of document reconstruction can occur. The most sensitive piece of information obtained is the title of the document that is opened.

The system automatically uploads the participants log files to our server when required. This action runs over HTTP port 80 (the same channel as that used for web-browsing) and so does not introduce any potential for security weaknesses on the user's system by opening other ports. The software allows the participant to view all of the data that is being logged and uploaded in real-time.

Participation

Participation in this study will be on a voluntary basis. The volunteers will be asked to read and sign a consent form (copy attached for reference) before the software is installed. The consent form makes the volunteers aware of their rights. The unique rights for this study include: the right to withdraw from the study at any time by having the software un-installed, the right to have their logged data destroyed and excluded from the study and the right to view the data that we are logging.

The volunteer must either be the sole user of the computer, or permission will be obtained from all users of the computer, as the software records the actions of all people who log into it. For this reason we will not be installing the software onto any shared-machines (such as those in the IT-department's work rooms), but will be seeking volunteers who have the exclusive use of a computer, such as staff members and post-graduate students.

Volunteers will be made aware of the chance to participate via email. Recruitment will begin in the Computer Science department and will then move to other departments in the College of Engineering, such as Electrical and Computer Engineering. Departments in the colleges of Science and Business & Economics, will then be approached. If sufficient participants are not recruited from these colleges, we will attempt to obtain volunteers from the College of Arts and the School of Law. We may offer the software to other students who have their own computers at home, but we will still ensure the above pre-conditions are met. Researchers from outside the University of Canterbury who have shown interest in our work may also participate in this study.

The software will be distributed to participants with the knowledge that we may be monitoring their actions for up to six months. The actual length of time will be largely dependent on the number of participants recruited and the amount of data (which is directly proportional to the amount of application usage) that we receive from our volunteers.

At the conclusion of the study we will ensure that the software is removed from all of the participants computers.

Data Dissemination

Uploaded log files contain the user code and machine name of the participant. This will allow us to characterise each person's scrolling actions individually. Only myself and my supervisors will have access to this identifiable data. Results that are published from this study will have all forms of identification stripped to maintain anonymity.

Appendix B

Applications Surveyed for Document Navigation Tool Support

T^{WELVE} document navigation systems were surveyed for their support of the tools described in Chapter 3. These applications were chosen due to their widespread use and availability. Each application was tested using its default setup. The version specifications and sources of further information (including how to obtain the software) are summarised in Table B.1.

Application	Version	OS	Source
Adobe Reader 7	7.0.8	Windows	www.adobe.com/products/reader/
Emacs/Xemacs	21.4.1/21.5	Linux	www.gnu.org/software/emacs/ & www.xemacs.org/
Evince Document Viewer	0.6.0	Linux	www.gnome.org/projects/evince/
Internet Explorer	7.0.5730.11	Windows	www.microsoft.com/ie/
Microsoft Powerpoint 2003	11.8169.8172	Windows	www.microsoft.com/powerpoint/
Microsoft Word 2003	11.8169.8172	Windows	www.microsoft.com/word/
Mozilla Firefox	2.0	Windows	www.mozilla.com/firefox/
Open Office Impress	2.0.4	Linux	www.openoffice.org/product/impress.html
Open Office Writer	2.0.4	Linux	www.openoffice.org/product/writer.html
Vi/Vim	7.0	Linux	www.vim.org
Microsoft Visual Studio 2005	8.0.50727.762	Windows	www.microsoft.com/vstudio/
Xpdf	3.01	Linux	www.foolabs.com/xpdf/

Table B.1: Applications surveyed for document navigation tool support

Appendix C

AppMonitor Log File Syntax

THE AppMonitor log files follow a consistent structure to ease the parsing and analysing process. The BNF grammar in Listing C.1 describes this structure.

```

1 <LogfileLine> ::= <DateAndTime>: <Event>
2 <Event> ::= <InternalEvent> | <HookerEvent>
3 <DateAndTime> ::= <Date><Time>
4 <Date> ::= <Day>/<Month>/<Year>
5 <Day> ::= <Integer>
6 <Month> ::= <Integer>
7 <Year> ::= <Integer>
8 <Time> ::= <Hours>:<Minutes>:<Seconds>.<Milliseconds>
9 <Hours> ::= <Integer>
10 <Minutes> ::= <Integer>
11 <Seconds> ::= <Integer>
12 <Milliseconds> ::= <Integer>
13 <InternalEvent> ::= <AppMonitorEvent> | <DocumentState> | <Debug> | <LogSent> |
14 <DocumentProperties>
15 <AppMonitorEvent> ::= AppMonitorStarted | AppMonitorExit | EventOptionsChanged
16 <DocumentState> ::= <NewDoc> | <DeadDoc> | <ScrollbarSetChange> | <ScrollbarChange> |
17 <PageStatusChange> | <ZoomChange>
18 <Debug> ::= Debug <String>
19 <LogSent> ::= ResumeAfterSendingFile <Filename>
20 <DocumentProperties> ::= DocumentProperties <WindowHandle> (<WindowHandle>) <String>
21 <ViewType> <Zoom>
22 <NewDoc> ::= NewDocument <WindowHandle> (<WindowHandle>) <ApplicationName>
23 <ScrollbarStatusList>
24 <DeadDoc> ::= DeadDocument <WindowHandle> (<WindowHandle>)
25 <ScrollbarSetChange> ::= ScrollbarSetChanged <WindowHandle> (<WindowHandle>)
26 <ScrollbarStatusList>
27 <ScrollbarChange> ::= ScrollBarsChanged <WindowHandle> (<WindowHandle>)
28 <ScrollbarStatusList>
29 <PageStatusChange> ::= PageStatusChanged <WindowHandle> (<WindowHandle>) <Integer> of
30 <Integer>
31 <ZoomChange> ::= ZoomChanged <WindowHandle> (<WindowHandle>) <Zoom>
32 <ApplicationName> ::= [MicrosoftWord] | [AdobeReader]
33 <ScrollbarStatusList> ::= <ScrollbarStatus> | <ScrollbarStatus> <ScrollbarStatusList>
34 <ScrollbarStatus> ::= <ScrollbarID> <ScrollbarState>
35 <ScrollbarID> ::= VE | VC | VU | VL | VX | HS | HC | X | VT | HT | VB | HB | VR | HR
36 <ScrollbarState> ::= (<ScrollbarMin>,<ScrollbarMax>,<ScrollbarStatic>,<Thumbsize>,<ScrollbarDynamic>)
37 <ViewType> ::= UnknownView | NormalView | PrintLayoutView | OutlineView | ReadingLayoutView
38 | WebLayoutView | SinglePageView | ContinuousView | ContinuousFacingView |
39 FacingView
40 <Zoom> ::= <PercentToken>
41 <HookerEvent> ::= <HookerEventCode> <WindowHandle> | <HookerEventCode> <WindowHandle>
42 <EventInformation>
43 <HookerEventCode> ::= EVENT_SYSTEM_SOUND | EVENT_SYSTEM_ALERT | ... |
44 EVENT_OBJECT_ACCELERATORCHANGE
45 <WindowHandle> ::= <HexadecimalInteger>
46 <ScrollbarMin> ::= <Integer>
47 <ScrollbarMax> ::= <Integer>
48 <ScrollbarStatic> ::= <Integer>
49 <Thumbsize> ::= <Integer>
50 <ScrollbarDynamic> ::= <Integer>
51 <EventInformation> ::= <Name> | <Name> / <Name>
52 <Name> ::= <String>
53 <Filename> ::= <String>
54 <PercentToken> ::= <Integer> %
55 <Integer> ::= <Digit> | <Digit> <Integer>
56 <Digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
57 <String> ::= A token delimited by double-quotes, e.g. "Hello world"
58 <HexadecimalInteger> ::= a token that represents a hexadecimal number, e.g. 0x125a4fc

```

Listing C.1: BNF syntax for AppMonitor log files

Appendix D

Longitudinal Study Material

D.1 Information Sheet

The information sheet that follows was provided to participants before they signed a consent form that indicated they agreed to participate in the longitudinal study.

Document Navigation Study Participant Information Sheet

We are carrying out an investigation into how people navigate within documents. In particular we are interested in the navigation actions that you perform within Microsoft Word and Adobe Reader. To do this we have written a program called AppMonitor. This software automatically logs navigation events that you produce in either of these two applications. Examples of these events include:

- A document being opened or closed.
- Scrollbar movement: How far the scrollbar was moved and what caused the move (a click on the arrows, thumb dragged, etc).
- Items being selected from menus.
- Window actions: The resizing, moving, minimising and maximising of the application window.
- Special key presses: Page up, Page down, the arrow keys, Tab, Enter, the Function keys (F1–F12) or any key press *with a modifier* (Ctrl or Alt held down simultaneously with a standard key).

Note that the software *does not* record every key-press entered into these applications. The most sensitive piece of information that may be recorded is the name of the document that you are working on. Once installed on your computer the AppMonitor program will start every time you log in (see note below about turning it off). We will leave the software installed on your computer for between three and six months (we will contact you when the study has been completed).

This study is in no way a test of your competence with computers or document navigation systems.

Thank-you for agreeing to participate in this study. If you have any questions about this investigation or any issues with the software, please contact us:

Jason Alexander (Ph.D. student)	jason@cosc.canterbury.ac.nz	+64 3 366 7001 x7755
Andy Cockburn (supervisor)	andy@cosc.canterbury.ac.nz	+64 3 366 7001 x7768

Please complete and sign the form on the following page *before* the software is installed on your computer.

Additional Information

If you wish to temporarily turn AppMonitor off, you can double click on the AppMonitor icon in the system tray (see figure 1) and choose File→Exit. It can be restarted later by selecting it from the Start Menu. More information about AppMonitor and how it works can be found at:
<http://www.cosc.canterbury.ac.nz/~jma142/AppMonitor/>



Figure 1: AppMonitor system tray icon

D.2 Consent Form

The consent form that follows was signed by all participants before AppMonitor was installed on their computer, at the beginning of the longitudinal study.

Document Navigation Study Consent Form

I consent to act as a participant in a study that will record document navigation actions that I perform in Microsoft Word and Adobe Reader. I agree to let the resulting data be used for analysis and presentation, subject to the conditions below:

- Only Jason and Andy will know the identity of the participant and their data;
- Data published or presented will be stripped of my identity;
- I retain the right to stop my role as a participant at any time without question, have the App-Monitor software un-installed and my data discarded.

I have read and understood the *Document Navigation Study Participant Information Sheet*.

Name:

Usercode:

Computer Name:

Email Address:

Signature:

Date:

D.3 Demographic Information Form

The demographic information form that follows was completed by all participants in the longitudinal study immediately after signing the consent form.

Document Navigation Study Questionnaire

Please complete this brief questionnaire at the beginning of the document navigation study.

1. What is your gender? Female ☐ Male ☐
2. What is your age? _____
3. Are you: Left-handed ☐ Right-handed ☐
4. Which hand do you control your computer mouse with? Left hand ☐ Right hand ☐
5. Does your computer have a mouse with a scroll wheel? Yes ☐ No ☐
6. What is your occupation? (if employed by, or studying at a university, please indicate department)
Occupation: _____
Department: _____
7. Approximately how many hours per week do you spend using a computer?
0–5 ☐ 5–10 ☐ 10–20 ☐ 20–30 ☐ 30–40 ☐ 40–50 ☐ 50+ ☐
8. Approximately what percent of these hours are for work (as opposed to pleasure)?
_____ %
9. I would describe my computer skills as (tick the box that you think describes you best):
☐ Beginner: Just learning how to use a word processor, email and browse the Internet.
☐ Intermediate: Can use a word processor, email and the Internet for simple tasks, but requires guidance on more difficult tasks.
☐ Confident: Regularly and confidently use computers for word processing, email, Internet browsing and possibly other tasks.
☐ Advanced: A confident user who has some knowledge about, or is learning to, program or administer computers.
☐ Expert: An experienced system administrator or computer programmer.
10. Approximately how often do you use Microsoft Word?
☐ Never
☐ Occasionally (once a week)
☐ Regularly (once a day)
☐ All the time (multiple times a day)
11. What do you use Microsoft Word for? (tick as many as appropriate)
☐ Never use it
☐ To read documents other people have created
☐ To edit or review documents other people have created
☐ To create my own documents
12. Approximately how often do you use Adobe Reader?
☐ Never
☐ Occasionally (once a week)
☐ Regularly (once a day)
☐ All the time (multiple times a day)

13. I believe I navigate (move) through documents effectively and efficiently (this includes when reading a document from start to end, creating new documents, searching in documents, etc):

Disagree ☐ ☐ ☐ ☐ ☐ Agree

14. I get a sore hand/wrist after navigating though large documents:

Disagree ☐ ☐ ☐ ☐ ☐ Agree

15. Use the space below to record any comments you have (positive or negative) about the tools you use for document navigation

Appendix E

Task-centric Observation Material

E.1 Consent Form

The consent form that follows was signed by all participants before beginning the task-study.

Document Navigation Experiment Consent Form

I consent to act as a participant in an experiment that will require me to perform document navigation tasks in Microsoft Word and Adobe Reader. The experiment software will record all actions that I perform, including scrolling actions, mouse movement and key presses. I agree to let the data gathered during this experiment to be used for analysis and presentation, subject to the conditions below:

- Only Jason and Andy will know the identity of the participant and their data;
- Data published or presented will be stripped of my identity;
- I retain the right to stop my role as a participant at any time without question, and have my data discarded

Name: _____

Signature: _____

Date: _____

Thank-you for agreeing to participate in this experiment. If you have any questions about this study please contact us:

Jason Alexander (Ph.D. student)	jason@cosc.canterbury.ac.nz	+64 3 364 2987 x7755
Andy Cockburn (supervisor)	andy@cosc.canterbury.ac.nz	+64 3 364 2987 x7768

E.2 Instruction Sheet

The instruction sheet that follows was displayed on-screen and verbal communicated to participants before beginning the task-study.

Document Navigation Experiment Instruction Sheet

- This experiment will ask you to complete a series of tasks in Microsoft Word and Adobe Reader.
- Please use these applications as you would normally, completing the tasks as quickly and as accurately as possible.
- The correct document and application will automatically open when the **Start Task** button is pressed.
- The document and application will automatically close when the **End Task** button is pressed.
- Please then write the answer to the task question in the answer box (if required).
- Answers are short (usually only one or two words or numbers).
- You cannot return to the document once the **End Task** button has been pressed, so you will need to memorise the answer for a few seconds.
- To move to the next task, click the **Next Task >>** button.
- The software will automatically monitor your actions (such as scrolling method, mouse position etc).
- To begin, click the **Next Task >>** button

E.3 Demographic Information Form

The demographic information form that follows was completed by all participants in the task-study immediately after signing the consent form.

Document Navigation Experiment Questionnaire

Please complete this brief questionnaire at the end of the document navigation experiment.

1. What is your gender? Female ☐ Male ☐
2. What is your age? _____
3. Are you: Left-handed ☐ Right-handed ☐
4. Which hand do you control your computer mouse with? Left hand ☐ Right hand ☐
5. Approximately how many hours per week do you spend using a computer?
0–5 ☐ 5–10 ☐ 10–20 ☐ 20–30 ☐ 30–40 ☐ 40–50 ☐ 50+ ☐
6. Approximately what percent of these hours are for work (as opposed to pleasure)?
_____ %
7. On average, approximately how often do you use Microsoft Word?
☐ Never
☐ Very Occasionally (at the most, once a month)
☐ Occasionally (at the most, once a week)
☐ Regularly (at the most, once a day)
☐ Frequently (multiple times a day)
8. What do you use Microsoft Word for? (tick as many as appropriate)
☐ Never use it
☐ To read documents other people have created
☐ To edit or review documents other people have created
☐ To create my own documents
9. On average, approximately how often do you use Adobe Reader?
☐ Never
☐ Very Occasionally (at the most, once a month)
☐ Occasionally (at the most, once a week)
☐ Regularly (at the most, once a day)
☐ Frequently (multiple times a day)
10. I found the tasks easy to complete when using the short document in Microsoft Word.

Disagree ☐ ☐ ☐ ☐ ☐ Agree
11. I found the tasks easy to complete when using the long document in Microsoft Word.

Disagree ☐ ☐ ☐ ☐ ☐ Agree
12. I found the tasks easy to complete when using the short document in Adobe Reader.

Disagree ☐ ☐ ☐ ☐ ☐ Agree
13. I found the tasks easy to complete when using the long document in Adobe Reader.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

14. What interface tools did you find useful when completing these tasks in Microsoft Word?

15. What interface tools did you find useful when completing these tasks in Adobe Reader?

16. What aspects of the Microsoft Word interface hindered you when trying to achieve these tasks?

17. What aspects of the Adobe Reader interface hindered you when trying to achieve these tasks?

Appendix F

Interactive Session Material

F.1 Consent Form

The consent form that follows was signed by all participants before beginning an interactive session.

Post-AppMonitor Study Interactive Session Consent Form

I consent to act as a participant in an interactive session that will require me to perform navigation tasks in Microsoft Word and Adobe Reader. The interactive session will be video recorded to allow further analysis to be performed at a later date. I agree to let the data gathered during this interactive session to be used for analysis and presentation, subject to the conditions below:

- Only Jason and Andy will know the identity of the participant and their data;
- Data published or presented will be stripped of my identity;
- I retain the right to stop my role as an interviewee at any time without question, and have my data and video recordings discarded

Name: _____

Signature: _____

Date: _____

Thank-you for agreeing to participate in this interactive session. If you have any questions about this interactive session please contact us:

Jason Alexander (Ph.D. student)	jason@cosc.canterbury.ac.nz	+64 3 364 2987 x7755
Andy Cockburn (supervisor)	andy@cosc.canterbury.ac.nz	+64 3 364 2987 x7768

F.2 Interviewer Instruction and Question Script

The instruction and question script that follows was used by the interviewer when running the interactive sessions.

Post-AppMonitor Study Interactive Session

Interviewer Instructions and Response Sheet

Jason Alexander

Participant:

1 Pre-session

1. Bring consent form, resources page, proof-reading document, usb key, video camera.

2 Introduction

1. Thank-you for agreeing to be part of this interactive session. I am interested in understanding how people use document navigation systems. We have collected much empirical data from the AppMonitor tool, the idea of this interactive sessions is to understand the reasons particular tools are selected for use.
2. I will be asking you to perform a series of simple tasks firstly using Microsoft Word and then Adobe Reader.
3. I wish to video tape the session so that I can review your actions at a later date.
4. [Give participant consent form]
5. [Start video recorder]
6. I will be watching how you perform the task and will ask you *why* you performed it in that way. This is not because the task was performed incorrectly, but because I am trying to understand why certain tools are used.
7. Please give as honest answers as possible, answers such as: “It is the only way I know how”, “that’s how I always do it” or “force of habit” are perfectly fine. If you do not know how to complete a question that is fine, just say so.
8. There are no right and wrong methods for completing these tasks.
9. Make yourself comfortable, with the mouse, keyboard and seat.
10. This first activity is an example, so that you get the idea of the format of the rest of the interactive session.
11. Please open the calculator application (Start → All Programs → Accessories → Calculator)
12. Use the calculator to determine $23 \div 17$
13. Why did you use the keys/mouse instead of the keys/mouse?
14. Close the calculator program
15. Now for the real questions

3 Microsoft Word

1. Can you please open Eval-Doc-3.doc?

[Method of opening]

☐ File → Open

☐ Toolbar button

☐ From folder

[Selection method]

☐ Double click

☐ 'Open' push-button

☐ Other:

2. (For participants in mock-up) I would now like you to make any changes to the interface so that is similar to that you would use everyday. For example if there are toolbars or panels you always have displayed, please set them up, also re-size the window to how you would normally use the program. You are free to change any of these settings at any time during the study.

[List any interface changes]

3. How many pages are there in this document?

[Answer: 9, correct: ☐ incorrect: ☐]

[Method of determining number of pages and why, any other way?]

[Show participant page counter if unaware]

4. [Show participant image]

5. What is the word highlighted in the red box in this image (note the text is upside down)?

[Answer: Canal, correct: ☐ incorrect: ☐]

[Zoom method and reason for using]

6. (?) Can you please return the document to a zoom that allows us to see the complete page width?

[Method of zooming out to page width view and reason]

7. Section 5.6 "Turn taking" contains several subsections, indicated by bold text. Please read out the title of each of these subsections.

☐ Ownership through Awareness

☐ Interruptions

☐ Assistance

☐ The Mode problem

[Navigation method and reason for using]

8. What *subsection* (e.g 5.1, 6.2) does the paragraph entitled “Gestures as consequential communication” belong to?

[Answer: 2.2 - Group Benefits, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

9. Can you please open Eval-Doc-5.doc?

[Method of opening] ☐ File → Open ☐ Toolbar button ☐ From folder
[Selection method] ☐ Double click ☐ ‘Open’ push-button ☐ Other:

10. [Show participant image]

11. What is the figure number under this image?

[Answer: 3, correct ☐ incorrect: ☐]

[Navigation method and reason for using]

12. What is the page number of the blank page (containing only the header and footer) in this document?

[Answer: 19, correct ☐ incorrect: ☐]

[Navigation method and reason for using]

13. On what page does the section called “Resources” start?

[Answer: 35, correct ☐ incorrect: ☐]

[Navigation method and reason for using]

14. What is the name of the first non-contents section where the phrase “user space” occurs?

[Answer: “Virtual Address Space”, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

15. What is the text in the heading at the top of page 13?

[Answer: "Memory Descriptor Lists", correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

16. Can you please return to the start of the document?

[Navigation method and reason for using]

17. How many times does the phrase "driver verifier" occur in this document?

[Answer: 8, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

18. What company do all of the websites listed on the last page of the document belong to?

[Answer: Microsoft, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

19. Can you please open Eval-Doc-4.doc?

[Method of opening] ☐ File → Open ☐ Toolbar button ☐ From folder

[Selection method] ☐ Double click ☐ 'Open' push-button ☐ Other:

20. How many boxes containing red text are there in this document?

[Answer: 4, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

21. [Give participant proof-read document with changes to be made]

22. Can you please open Eval-Doc-6.doc?

[**Method of opening**] ☐ File → Open ☐ Toolbar button ☐ From folder

[**Selection method**] ☐ Double click ☐ 'Open' push-button ☐ Other:

23. Please make the changes as shown in the comments in this proof-read document. Save the document when you have finished.

[**Navigation method and reasons for each correction**]

[**Change 1: 1975 → 1985**]

[**Change 2: Delete "Visual Arts"**]

[**Change 3: Move "Design Arts"**]

[**Change 4: "5" → "Five"**]

[**Change 5: "were asked to complete" → "completed"**]

[**Change 6: "within" → "by"**]

[**Change 7: Delete sentence**]

[**Change 8: "44%" → "54%"**]

[**Change 9: "233" → "234"**]

[Method of saving and reason]

24. Can you please open Eval-Doc-7.doc?

[Method of opening] ☐ File → Open ☐ Toolbar button ☐ From folder
[Selection method] ☐ Double click ☐ 'Open' push-button ☐ Other:

25. A copy of the first paragraph of the introduction has been modified and placed at the end of the introduction section (named "Introduction (changed)"). What is the difference between these two introduction paragraphs?

[Answer: The phrase in brackets "(i.e., it is intended for all users and retrieval of all types of information)" has been removed in the second version (in the third sentence).]

[Navigation method and reasons]

26. In this section I have a small subset of the more unique tools in Microsoft Word that I am interested in finding out if you are familiar with. For each tool I will get you to demonstrate it, if you know how, otherwise simply let me know that you are not familiar with it.

Microsoft Word Competency and Frequency Chart

Tool	Demonstrated Competency										I frequently use this tool										Comments
Rate-based Scrolling	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Zoom control using "Ctrl-Scrollwheel"	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Thumbnail Pane	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Document Map	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Split Views of the same document	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
"Browse by..." (page, section, comment etc)	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	

Adobe Reader Competency and Frequency Chart

Tool	Demonstrated Competency										I frequently use this tool										Comments
Automatic Scroll function	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Dynamic Zoom	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
First Page/Last Page functions	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Previous view/Next view functions	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Zoom control using Ctrl-scrollwheel	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Thumbnails tab	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Bookmarks tab	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Find functionality	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	
Search functionality	Not familiar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Competent	Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree	

4 Adobe Reader

1. We will now move onto tasks using Adobe Reader.
2. Can you please open Eval-Doc-3.pdf?
[**Method of opening**] ☐ File → Open ☐ Toolbar button ☐ From folder
[**Selection method**] ☐ Double click ☐ 'Open' push-button ☐ Other:
3. As per Microsoft Word, feel free to make any interface changes you wish to ensure the environment is similar to that you use everyday.
4. How many pages are there in this document?
[**Answer: 10, correct: ☐ incorrect: ☐**]
[**Method of determining number of pages and why, any other way?**]

[**Show participant page counter if unaware**]
5. [**Show participant image**]
6. The search box in this image contains a word - what is it?
[**Answer: "Investigate", correct: ☐ incorrect: ☐**]
[**Zoom method and reason for using**]

7. (?) Can you please return the document to a zoom that allows us to see the complete page width?
[**Method of zooming out to page width view and reason**]

8. The "Cheating" section contains five bullet points. Can you please read the titles of each of the bullet points (in bold) out to me in the order they appear.
☐ The player queue ☐ IP address checks ☐ Seed images
☐ Limited freedom to enter guesses ☐ Aggregating data from multiple players
[**Navigation method and reason for using**]

9. What *section* (in capital letters) does the subsection "Alternative: Using Ping Data for Pointing" belong to?

[Answer: “Additional Applications”, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

10. Could you please open Eval-Doc-5.pdf

[Method of opening]

☐ File → Open

☐ Toolbar button

☐ From folder

[Selection method]

☐ Double click

☐ ‘Open’ push-button

☐ Other:

11. [Show participant image]

12. What is the number of the figure in this image?

[Answer: 3, correct ☐ incorrect: ☐]

[Navigation method and reason for using]

13. What is the page number of the blank page in this document?

[Answer: 43, correct ☐ incorrect: ☐]

[Navigation method and reason for using]

14. On what page does the section called “Summary” start?

[Answer: 45, correct ☐ incorrect: ☐]

[Navigation method and reason for using]

15. What is the name of the section where the phrase “latent variables” first occurs?

[Answer: “Some Gaussian Process Models”, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

16. What is the text in the heading near the top of page 30?

[Answer: “Choice of Kernel. Kernel Design”, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

17. Can you please return to the start of the document?

[Navigation method and reason for using]

18. How many times does the phrase “Gaussian process model” occur in this document?

[Answer: 9, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

19. What is the most recent year listed on the last page of the document?

[Answer: 1999, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

20. Can you please open Eval-Doc-4.doc?

[Method of opening]

☐ File → Open

☐ Toolbar button

☐ From folder

[Selection method]

☐ Double click

☐ ‘Open’ push-button

☐ Other:

21. How many diagrams are there that are surrounded by a red box?

[Answer: 5, correct: ☐ incorrect: ☐]

[Navigation method and reason for using]

22. Can you please open Eval-Doc-7.doc?

[Method of opening]

☐ File → Open

☐ Toolbar button

☐ From folder

[Selection method]

☐ Double click

☐ ‘Open’ push-button

☐ Other:

23. A copy of the first paragraph of the “Reducing User Frustration” section has been modified and placed at the end of the second section (named “Reducing User Frustration (changed)”). What is the difference in the content between these two paragraphs?

[Answer: In the first paragraph the two items in the last sentence are listed as (1) and (2) and the second paragraph they are not.]
[Navigation method and reason for using]

24. In this section I have a small subset of the more unique tools in Adobe Reader that I am interested in finding out if you are familiar with. For each tool I will get you to demonstrate it, if you know how, otherwise simply let me know that you are not familiar with it (see table on page 7).

5 Participant Input

1. Now is your chance to tell me about what you would like to see in or what you currently think about document navigation systems.
2. Perhaps you could show me which program you would use if I gave you `ShakespearesSonnets.txt`. How would you navigate around this document?

3. Are there any features in the document navigation (or other) systems that you use that you would like to see in all document navigation systems?

4. What are systems do you use that involve documents (e.g writing code)?

5. Would you like to demonstrate that too me? Are there features in there you use regularly?

6. What frustrates you about document navigation systems?

7. Do you have any other comments about the AppMonitor study, this interactive session or document navigation in general?

6 Conclusion

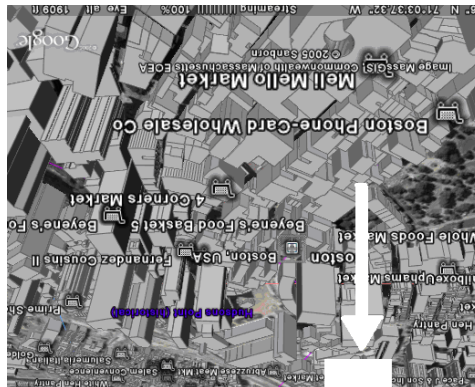
Thank-you for your time.

F.3 Interactive Session Resources

The resources sheet that follows was used to present additional information to participants during the interactive sessions.

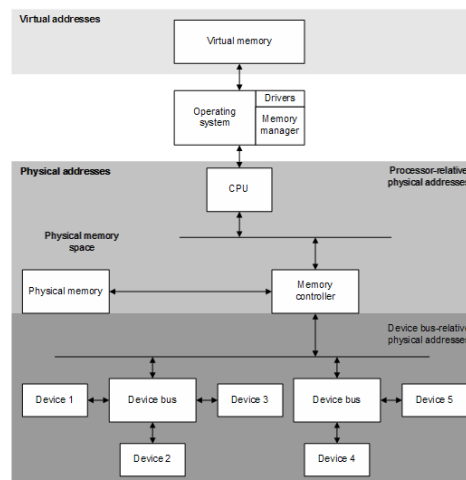
1 Microsoft Word Question Resources

Question 5: Identify the word in the red highlighted box



Question 8: “Gestures as consequential communication”

Question 11: What is the figure number under this image?

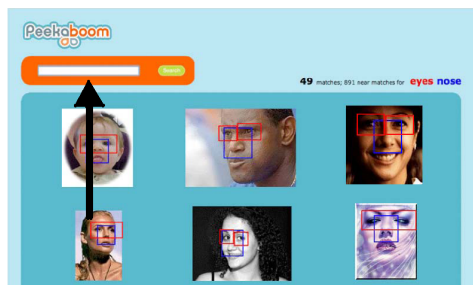


Question 14: “user space”

Question 17: “driver verifier”

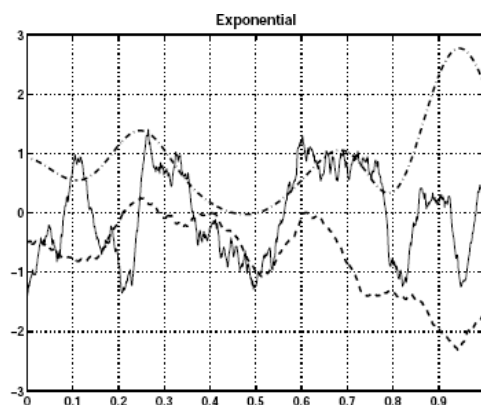
2 Adobe Reader Question Resources

Question 6: What is the word located inside the search box in the image below?



Question 9: “Alternative: Using Ping Data for Pointing”

Question 12: What is the figure number under this image?



Question 15: “latent variables”

Question 18: “Gaussian process model”

Appendix G

Footprints Scrollbar Material

G.1 Marking Scrollbar Evaluation Consent Form

The consent form that follows was signed by all participants before beginning the Marking Scrollbar evaluation.

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF SASKATCHEWAN
INFORMED CONSENT FORM



Research Project: **Finding text using readwear**

Investigators: Jason Alexander jason@cosc.canterbury.ac.nz
Carl Gutwin gutwin@cs.usask.ca
Andy Cockburn andy@cosc.canterbury.ac.nz

This consent form, a copy of which has been given to you, is only part of the process of informed consent. It should give you the basic idea of what the research is about and what your participation will involve. If you would like more detail about something mentioned here, or information not included here, please ask. Please take the time to read this form carefully and to understand any accompanying information.

This study is concerned with detecting whether or not markings on a scroll bar help you remember where particular text is located within a document.

The goal of the research is to determine whether markings on a scroll bar help participants remember where particular text is located within a document.

The session will require **30** minutes, during which you will be asked to find, and refer back to, passages from a text document.

At the end of the session, you will be given more information about the purpose and goals of the study, and there will be time for you to ask questions about the research.

The data collected from this study will be used in articles for publication in journals and conference proceedings.

As one way of thanking you for your time, we will be pleased to make available to you a summary of the results of this study once they have been compiled (usually within two months). This summary will outline the research and discuss our findings and recommendations. If you would like to receive a copy of this summary, please write down your email address here.

Contact email address: _____

All personal and identifying data will be kept confidential. If explicit consent has been given, textual excerpts, photographs, or videorecordings may be used in the dissemination of research results in scholarly journals or at scholarly conferences. Anonymity will be preserved by using pseudonyms in any presentation of textual data in journals or at conferences. The informed consent form and all research data will be kept in a secure location under confidentiality in accordance with University policy for 5 years post publication. Do you have any questions about this aspect of the study?

You are free to withdraw from the study at any time without penalty and without losing any advertised benefits. Withdrawal from the study will not affect your academic status or your access to services at the university. If you withdraw, your data will be deleted from the study and destroyed.

Your continued participation should be as informed as your initial consent, so you should feel free to ask for clarification or new information throughout your participation. If you have further questions concerning matters related to this research, please contact:

Your signature on this form indicates that you have understood to your satisfaction the information regarding participation in the research project and agree to participate as a participant. In no way does this waive your legal rights nor release the investigators, sponsors, or involved institutions from their legal and professional responsibilities. If you have further questions about this study or your rights as a participant, please contact:

- Jason Alexander, jason@cosc.canterbury.ac.nz
- Carl Gutwin, gutwin@cs.usask.ca
- Andy Cockburn, andy@cosc.canterbury.ac.nz
- Office of Research Services, University of Saskatchewan, (306) 966-4053

Participant's signature: _____

Date: _____

Investigator's signature: _____

Date: _____

A copy of this consent form has been given to you to keep for your records and reference. This research has the ethical approval of the Office of Research Services at the University of Saskatchewan.

G.2 Electronic Demographic Information Form

The first two experiments conducted using the Marking Scrollbar and the third experiment using the Footprints Scrollbar electronically collected demographic information from the study participants. The form used for this purpose is shown in Figure G.1.

G.3 Electronic Subjective Evaluation Form

The first two experiments conducted using the Marking Scrollbar and the third experiment using the Footprints Scrollbar electronically collected subjective evaluation information from the study participants. The form used for this purpose is shown in Figure G.2.

G.4 Footprints Scrollbar Evaluation Consent Form

The consent form that follows was signed by all participants before beginning the Footprints Scrollbar evaluation.

Demographic Information

Please use this form to enter demographic information about yourself.

Are you male or female?

☒ Male ☐ Female

What is your age, in years?

Are you left or right handed?

☐ Left Handed ☐ Right Handed

What is your occupation (if student or professor, please note subject area)?

What is your approximate computer use, in hours, per week?

☐ 0-10 ☐ 10-20 ☐ 20-30 ☐ 30-40 ☐ 40-50 ☐ 50-60 ☐ 60-70 ☐ 70-80 ☐ 80+

Please rank the five activities that account for the greatest amount of the time you spend using a computer .

(e.g. email, internet browsing, document creation, document reading, writing code, gaming, accounting etc)

1

2

3

4

5

OK

Figure G.1: Electronic demographic information form

Subjective Evaluation

Please use this form to enter your feedback on the system you have just used.

Mental demand

How much mental and perceptual activity was required (e.g. thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy or demanding, simple or complex, forgiving or exacting?

Low High

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Physical demand

How much physical activity was required (e.g., pushing, pulling turning, controlling, activating, etc.)? Was the task easy or demanding, slow or brisk, slack or strenuous, restful or laborious?

Low High

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Temporal demand

How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

Low High

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Performance

How successful do you think you were in accomplishing the goals of the task set by the experimenter? How satisfied were you with your performance in accomplishing these goals?

Poor Good

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Frustration level

How secure, gratified, content, relaxed and complacent versus insecure, discouraged, irritated, stressed and annoyed did you feel during the task?

Low High

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Effort

How hard did you have to work (mentally and physically) to accomplish your level of performance?

Low High

☐ 1 ☐ 2 ☐ 3 ☐ 4 ☐ 5

Please use the following textbox to make any comments about the system you have just used:

OK

Figure G.2: Electronic subjective evaluation form, with NASA-TLX questions and freeform comment area.

Footprints Scrollbar Evaluation Consent Form

I consent to act as a participant in an experiment that will require me to navigate to specified document positions, using two different systems. The experimental interface will record actions that I perform. I agree to let the data gathered during this experiment to be used for analysis and presentation, subject to the conditions below:

- Only Jason and Andy will know the identity of the participant and their data;
- Data published or presented will be stripped of my identity;
- I retain the right to stop my role as a participant at any time without question, and have my data discarded

Name: _____

Signature: _____

Date: _____

Thank-you for agreeing to participate in this experiment. If you have any questions about this study please contact us:

Jason Alexander (Ph.D. student)	jason@cosc.canterbury.ac.nz	+64 3 364 2987 x7755
Andy Cockburn (supervisor)	andy@cosc.canterbury.ac.nz	+64 3 364 2987 x7768

G.5 Footprints Scrollbar Evaluation Feedback Form

The feedback form that follows was completed by all participants at the conclusion of the Footprints Scrollbar evaluation.

Standard Scrollbar Questionnaire

1. This interface was effective for completing the given tasks

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

Footprints Scrollbar Questionnaire

General

1. This interface was effective for completing the given tasks

Disagree ☐ ☐ ☐ ☐ ☐ Agree

2. I used the features of the Footprints Scrollbar to help me relocate positions within the document that I had already seen.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

3. I could easily predict when marks would be added and removed from the scrollbar.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

4. I would use the Footprints Scrollbar system if it was added to my everyday document navigation applications.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

Footprints Scrollbar Features

5. I found the marks in the scrollbar useful when completing the tasks.

Disagree ☐ ☐ ☐ ☐ ☐ Agree or ☐ Didn't use

6. I found the thumbnail previews useful when completing the tasks.

Disagree ☐ ☐ ☐ ☐ ☐ Agree or ☐ Didn't use

7. I found the numerical shortcuts useful when completing the tasks.

Disagree ☐ ☐ ☐ ☐ ☐ Agree or ☐ Didn't use

8. I found the back/forward arrow keys useful when completing the tasks.

Disagree ☐ ☐ ☐ ☐ ☐ Agree or ☐ Didn't use

9. I found the Shift-scrollwheel technique useful when completing the tasks

Disagree ☐ ☐ ☐ ☐ ☐ Agree or ☐ Didn't use

10. It was easy to confuse the meaning of the colours and numbers in the scrollbar.

Disagree ☐ ☐ ☐ ☐ ☐ Agree or ☐ Didn't use either

Comments:

Comments

11. What part(s) of the Footprints Scrollbar did you find most useful?

12. What part(s) of the Footprints Scrollbar did you find distracting, confusing or not useful?

13. Do you have any other comments about the Footprints Scrollbar system?

Overall

Please indicate overall which interface you preferred for completing these tasks:

☐ Standard Scrollbar ☐ Footprints Scrollbar

G.6 Footprints Scrollbar Observational Study Interviewer Instruction Sheet

The instruction sheet that follows was used by the interviewer during the observational study. It includes the full list of questions posed to the participants.

Interviewer Instructions for the Footprints Scrollbar Observational Study

Please tell me your favourite research paper published in a conference or journal, either your own or someone else's.

[Retrieve the paper and convert it for use in the Footprints system]
[With the demonstration paper in the system]

This is our Footprints Scrollbar. It is designed to help you revisit regions you've recently seen in your document. It has five different tools for to aid quick revisitation:

1. Up to ten coloured marks are placed in the scrollbar to show places you've visited. Marks gradually fade from 'hot' colours (reds, oranges) through to 'cold' (greens, blues) to denote their increasing age. Importantly, marks are only placed when you pause in a spot for more than two seconds, and you know when marks are placed because the scroll thumb gradually 'fills' and marks are placed once full.
2. You can right click on marks to go to them quickly.
3. Marks are overlaid with numbers 1–10, and these act as shortcuts—just type the number to go there. The numbers don't change, but they are reused: so the first place you visit will be item 1, and it'll stay item 1 until it disappears off the end of the recency list, at which point the newly added item becomes item 1.
4. The 'left' and 'right' keyboard keys allow you to move through the mark history like 'Back' and 'Forward' buttons on the web.
5. Small thumbnail images appear when you put the cursor over the scrollbar, and they zoom further when you mouse over them. You can click them to go there.

[Instruct participants to experiment with all of the tools (tick them off one by one) in the demonstration document]

[Load converted paper into the Footprints system]

Here's your document. I now have a series of questions about your paper that I'd like you to answer. Feel free to make use of all, some or none of the additional features that the Footprints Scrollbar provides.

1. Show me what you think is the best part of the paper
2. Who's on the reference list?
3. Where's related work summarised: at the start, at the end or both?
4. What's the first paragraph of the Introduction?"
5. Where are the main details of the research method?
6. And the main results?
7. Show me the references again?
8. What's the first paper referenced in the Introduction?
9. Where was that paper published?
10. Can you show me the research method again?

11. Show me the bit you think is the best again?
12. What's the first paper referenced from the related work?
13. Where was it published?
14. Is there a figure summarising the results?
15. Do any of the results feature in the abstract?
16. Are the Conclusions similar to the abstract?
17. Do the Conclusions restate/summarise information in the main results figure?
18. Which parts of the 'best bit' feature in the conclusions?
19. Which parts of the 'best bit' feature in the abstract?
20. The introduction once more?
21. Related Work?
22. Conclusions?
23. Results?
24. Introduction?
25. Results?

G.7 Footprints Scrollbar Observational Study Scenario Questionnaire

The questionnaire that follows was completed by all participants at the conclusion of the observational study.

Footprints Scrollbars Observational Study Questionnaire

Marks

1. Scrollbar marks were helpful in seeing where I had been.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

2. Scrollbar marks were distracting when I wanted to go somewhere I hadn't been before.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

3. I understood why marks were placed where they were.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

4. The 2 second timeout for mark placement was:

Too short ☐ ☐ ☐ ☐ ☐ Too long

Good

Comments:

5. The mark colours were useful:

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

6. Right clicking on marks/thumbnails was useful:

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

Mark Shortcuts

7. The shortcut numbers on marks were useful.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

8. The shortcut numbers on marks were distracting.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

Back/Forward Arrows

9. The Back/Forward Arrows were useful for revisitation.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

10. I understood the behaviour of the Back/Forward Arrows.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

Thumbnails

11. The thumbnail images were useful for revisitation.

Disagree ☐ ☐ ☐ ☐ ☐ Agree

Comments:

General

12. Rank (from 1 best to 5 worst) the usefulness of the following navigation methods:


Dragging to regions marks:	_____	Useful? Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree
Clicking on region marks:	_____	Useful? Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree
Mark number shortcuts:	_____	Useful? Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree
Back/Forward arrows:	_____	Useful? Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree
Clicking on thumbnails:	_____	Useful? Disagree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Agree

13. Please comment on the positive aspects of the Footprints Scrollbar

14. Please comment on the negative aspects of the Footprints Scrollbar

15. Would you want some or all of these features in your desktop interfaces?

Some ☐ Which?



All ☐

None ☐

16. Comments
