Collaborative adaptive accessibility and human capabilities

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

• A Doctoral Thesis. Submitted in partial fulfillment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/10262](https://dspace.lboro.ac.uk/2134/10262)

Publisher: © Matthew Tylee Atkinson

Please cite the published version.
This item was submitted to Loughborough’s Institutional Repository (https://dspace.lboro.ac.uk/) by the author and is made available under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
Collaborative Adaptive Accessibility and Human Capabilities

by

Matthew Tylee Atkinson

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy

of

Loughborough University

29th February 2012

Copyright 2012 Matthew Tylee Atkinson
Abstract

This thesis discusses the challenges and opportunities facing the field of accessibility, particularly as computing becomes ubiquitous. It is argued that a new approach is needed that centres around adaptations (specific, atomic changes) to user interfaces and content in order to improve their accessibility for a wider range of people than targeted by present Assistive Technologies (ATs). Further, the approach must take into consideration the capabilities of people at the human level and facilitate collaboration, in planned and ad-hoc environments.

There are two main areas of focus: (1) helping people experiencing minor-to-moderate, transient and potentially-overlapping impairments, as may be brought about by the ageing process and (2) supporting collaboration between people by reasoning about the consequences, from different users’ perspectives, of the adaptations they may require.

A theoretical basis for describing these problems and a reasoning process for the semi-automatic application of adaptations is developed. Impairments caused by the environment in which a device is being used are considered. Adaptations are drawn from other research and industry artefacts. Mechanical testing is carried out on key areas of the reasoning process, demonstrating fitness for purpose.

Several fundamental techniques to extend the reasoning process in order to take temporal factors (such as fluctuating user and device capabilities) into account are broadly described. These are proposed to be feasible, though inherently bring compromises (which are defined) in interaction stability and the needs of different actors (user, device, target level of accessibility).

This technical work forms the basis of the contribution of one work-package of the Sustaining ICT use to promote autonomy (Sus-IT) project, under the New Dynamics of Ageing (NDA) programme of research in the UK. Test designs for larger-scale assessment of the system with real-world participants are given. The wider Sus-IT project provides social motivations and informed design decisions for this work and is carrying out longitudinal acceptance testing of the processes developed here.

Keywords: Accessibility, Adaptation, Collaboration, Capability, User Interface.
Acknowledgements

I’ve witnessed many friends finding this the most difficult part to write and this is not going to be an exception. Apologies for its inevitable incompleteness and inability to do justice to the tremendous support I have received from so many in getting to this point.

Special thanks go to the NHS; the paramedics and staff of Leicester Royal Infirmary, who put me back together after a major road traffic accident. Those responsible included: Mr Tim Green and Mr Andrew Furlong, my consultants; the vascular surgeon who helped save my left leg; the people in the Emergency Department, ITU\(^1\) and on wards 17, 31 and 32; the British Red Cross, for support that was vital to my return to work; as well as all of the other doctors; nurses; physio therapists; occupational therapists; radiologists; neuropsychologists; social workers and medical secretaries that were involved in my rehabilitation. Though it is beyond the scope of this document to name them all, they provide support that is absolutely vital, far too often goes quietly un-noticed and has given me a second chance at life.

Let’s not forget the person in charge of the car, particularly his insurers, for sponsoring my period of action research—experience that has further reinforced to me the need to comprehensively deal with multiple impairments of a varied nature. Without the dedicated and incredibly reassuring support that can only arise from first-hand experience and a genuine desire to help people in some of the most difficult circumstances, provided by my solicitor, Paul Darlington, my family would have been completely lost and I shudder to think where I would have ended up. One of my carers (whom were all excellent and organised through the university and CSV charity) was called Paul, as was my councillor, so even though the driver was too, we are still in positive Paul territory, which is perhaps almost entirely inconsequential.

Of course it was actually my family (though they’ll be dealt with shortly), friends and colleagues who had the biggest and most positive impact on my return to work and the research as a whole. Without their support whilst I was in hospital

\(^1\)Prem: thanks for putting up with the psychedelic rants about the regrettable shift to running Windows over OS/2 on hospital equipment!
and help when I got out, including carting me around to various social outings, things would never have worked. They have provided a source of support I could never have expected and that gave me the determination to “KBO”. Most notable mentions must go to Mark Withall, John Whitley, Aaron Parker, Iain Phillips, Jeff Fry, Tim Verlander, Alice Ginniff, Alexandra Alecu, Lezan Hawizy, Scott Culcheth, Becky Yates, Kushvinder Gill, Wendy Olphert and Leela Damodaran. All of these people have given up a great deal of their time to give me advice and support, as well as a particularly entertaining office and home experience. My compatriots in more recent years have been equally supportive and kept being in the office engaging and sometimes far too amusing; my deepest thanks to Christine Bagley, Wendy Ng, Kirsty Smith, Matthew Bell and Terri Gilbertson.

My supervisor was alerted to the start-up of a preparatory network which would ultimately become the Sus-IT project. I have been honoured to be involved from the very start and help shape the bid in the technological department and then see the project through. The diversity and determination of the people I have met through Sus-IT—participants and researchers alike—has been phenomenal. I have also been fortunate enough to have the opportunity to teach and even produce some material for a module two years running whilst I have been at work; this was not only the most rewarding professional experience I have yet had, but it also gave me a great deal of experience and motivation to use in my research and in life in general. I must thank Roger Stone, who was with me from the start as my first-year undergraduate personal tutor, and Ray Dawson, for having faith in me to do a good job for the students. Roger was, as ever, an excellent mentor in this.

We return to my supervisor, Colin Machin; he has stuck by me through thick and thick and gone above and beyond to support me and enable me to take opportunities to better myself. Many thanks to him and of course Mandy Sherrington for providing some great evenings out and more encouragement to see this through. In my first year I was also incredibly lucky to have the guidance of Adrian Lawrence, which helped get me into research in the first instance, via my final-year project, and opened my eyes as to its importance and interest to me. Another major benefactor from those early days was the Grundy Educational Trust—my particular thanks to the trustees and Alicia Hardy—without this financial support I could never have even begun my course.

I have no idea how to finish this; this seems to be a perennial problem of mine. Not a day goes by when I don’t consider how incredibly lucky I am to be in the position of having such a “problem”. These years have actually been some of the best I’ve had, despite the difficulties. I have done so much, seen so much and learnt so much about myself and others that I couldn’t even have imagined
before I started. Independence; conferences; incredibly rewarding work and the vast majority of it has been incredibly fulfilling.

I have seen many contemporary examples of great courage and determination. John Whitley and my cousin Nicola Robinson provided me with two prime bits of motivation; they have both overcome adversity in different but equally awe-inspiring ways. The determination of friends like Alex, Lezan, Senay and Christine likewise.

In closing, however, I dedicate this thesis to two people who’ve shown love, wisdom, courage and patience in amounts to which I can only aspire, and will never fully appreciate or understand—my parents: Jean and Robin. You have given me everything that makes me who I am, and I can only hope that by finding the courage to see this through, I have done what I can to honour that.
# Contents

Abstract ii

Acknowledgements iii

1 Introduction 1
   1.1 Overview of Technical Motivations 2
   1.2 Contributions 2
   1.3 Thesis Structure 3

2 Literature Review 5
   2.1 Defining Usability 5
   2.2 Adaptive Interfaces 6
      2.2.1 Behavioural Classification 6
      2.2.2 Architecture 8
      2.2.3 Scope of Adaptations 10
   2.3 Critique of Adaptive User Interfaces 11
      2.3.1 Case Study: A Commercial Adaptive Interface 13
      2.3.2 Adaptability 14
   2.4 Ubiquitous Computing and Abstract Interfaces 15
      2.4.1 The “Everyday Computing” Vision 15
      2.4.2 Renewed Need for Abstraction 16
      2.4.3 Generality 17
      2.4.4 Criticisms 18
      2.4.5 Collaboration 18
      2.4.6 Accessible Collaboration Case Study 19
   2.5 Defining Accessibility 20
      2.5.1 Broad Definitions 21
      2.5.2 Public Perception 21
      2.5.3 Accessibility in terms of Barriers 21
   2.6 Defining (Computer Interface) Accessibility Barriers 22
      2.6.1 Scope of these Barriers 23
2.7 Universal Design ........................................... 24
2.8 A Spectrum of Capabilities ................................ 25
  2.8.1 Adaptive Accessibility Features in Computer Systems .... 26
  2.8.2 The Notion of Disability ................................ 26
2.9 Contrasting Accessibility and Usability ...................... 28
  2.9.1 Causal Relationships ................................. 28
  2.9.2 Effects on the Population ............................. 30
  2.9.3 Usability Problems vs. Accessibility Barriers .......... 31
  2.9.4 Definitions Used by this Thesis ..................... 32
2.10 Present Assistive Technology .............................. 33
  2.10.1 Availability ........................................ 34
  2.10.2 Standards ........................................ 35
  2.10.3 Unanticipated Effects .............................. 36
2.11 Problems with Present Assistive Technology .............. 37
  2.11.1 Undiscoverability and Immobility .................... 37
  2.11.2 Indirection ....................................... 38
  2.11.3 Rigidity of Underlying Systems ..................... 39
  2.11.4 Low Recognition of Dynamic Diversity ............... 40
  2.11.5 Inconsistency .................................... 41
  2.11.6 Lack of Accessibility in Authoring Tools .......... 42
  2.11.7 General Lack of Industry Adoption ................ 43
2.12 Adaptivity as a Means to Mainstream Accessibility ........ 43
  2.12.1 Variety and Configuration of Adaptations .......... 44
  2.12.2 Accepting, Exposing and Sensibly Applying Adaptations 45
2.13 Encouraging Industry Adoption .......................... 46
  2.13.1 Configuration and Security Matters ................ 47
  2.13.2 Criticisms of Adaptivity Revisited ................ 48
2.14 Summary ................................................ 49
  2.14.1 Commercial and Research Assistive Technology ........ 49
  2.14.2 Adaptivity for Accessibility ....................... 50
  2.14.3 General Adaptive Interfaces ....................... 52
  2.14.4 Distributed Interfaces ............................. 53
  2.14.5 Collaborative Interfaces ........................... 53
  2.14.6 Emerging Paradigms ................................ 54
2.15 Goals and Scope ........................................ 55
  2.15.1 Approach ................................ .......... 55
  2.15.2 Selected Technical Goals ............................ 57
  2.15.3 Sus-IT's Wider Goals ................................ 59
## 3 Approaches

3.1 Notable Approaches

- Technical
- Domain-specific vs. General
- Anticipated Long-term Developments

3.2 Theory Pre-requisites

- Frame of Reference: Conveying Information
- Defining and Measuring Capabilities
- Representing Capabilities, Barriers and Adaptations

3.3 The Capability-based Adaptive Accessibility Reasoning Approach

- Standards
- Deconstruct the Interface
- Domain-agnostic Decision-making
- Layers of Reasoning
- Reasoning in Action: Decision Trees
- Run-time Components

3.4 Relation to Thesis Goals

3.5 Application: the Sus-IT Project

- Potential Pitfalls, Criticisms and Justifications
- Infrastructure for Adaptation Discovery and Installation
- Interaction Workflow
- “Help Button” and Adaptation Suggestions
- Testing the Concept

3.6 Summary

## 4 Fundamental Reasoning Actors

4.1 Proposed Usability/Accessibility Indicator

- Rate and Scope
- Features of the Indicator

4.2 Relating the User, Adaptations, Devices and Capabilities

- Focusing on Information Transfer
- Classifying Capabilities and Components

4.3 Information and Device Characteristics

- Modalities and Data Types
- Multiplexing
- Organisation of Devices

4.4 Applications and Content

- Interface Tree Summary
- Applications, Data and Capabilities
## 5 Collaboration as Synchronisation

5.1 Synchronisation ........................................ 102
5.1.1 Relating Input, Focus and Devices ............... 102
5.2 Example Scenario ....................................... 105
5.2.1 Cursor Assignments ................................. 105
5.3 Defining Synchronisation .............................. 107
5.3.1 Coherence ............................................. 107
5.3.2 Interference .......................................... 108
5.3.3 Optimal Content and Alternatives ............... 108
5.3.4 Replacements as Coarse Adaptations .......... 109
5.4 Ideal Collaboration Solutions ....................... 110
5.4.1 Search Space and Technique ..................... 111
5.4.2 Constraints and Cost Factors ..................... 115
5.5 Collaboration Scenarios and Cost Factors .......... 116
5.5.1 Local Collaboration Variants ..................... 116
5.5.2 Remote User and Application Synchronisation .. 117
5.5.3 Combination of Cost Factors ...................... 117
5.6 Pragmatic Solution ..................................... 119
5.7 Redundancy and Replacements ....................... 121
5.8 Summary .................................................. 122

## 6 Adaptations and Capabilities

6.1 Overview ............................................... 123
6.1.1 Outline .............................................. 126
6.2 Real-world Reasoning .................................. 126
6.2.1 Ethical Considerations ............................. 126
6.2.2 Adaptation Initiative .............................. 127
6.2.3 General Problem Detection ....................... 127
6.3 Classifying Adaptations .............................. 128
6.4 Relation to Information Systems Theory .......... 129
6.4.1 Time .................................................. 131
6.5 Channels ............................................... 132
6.5.1 Channels and Capabilities ....................... 133
6.5.2 Capability Requirements as Bandwidth Limitations . 134
6.5.3 Mitigating Capability Gaps with Adaptations .... 135
6.6 Consequences and Abstract Adaptations .......... 136
6.6.1 Consequences of Adaptations .................... 139
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.1</td>
<td>Inherent to the CAAR Process</td>
<td>186</td>
</tr>
<tr>
<td>8.2</td>
<td>Classification, Indicators and Possible Adaptations</td>
<td>186</td>
</tr>
<tr>
<td>8.2.1</td>
<td>Indicators from Orthogonal Facets</td>
<td>187</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Comparing Aesthetics and Layout</td>
<td>189</td>
</tr>
<tr>
<td>8.3</td>
<td>Collaboration Set-up</td>
<td>190</td>
</tr>
<tr>
<td>8.3.1</td>
<td>Complete Search Performance</td>
<td>190</td>
</tr>
<tr>
<td>8.3.2</td>
<td>Seeded Search Performance</td>
<td>191</td>
</tr>
<tr>
<td>8.3.3</td>
<td>Capability-based Search</td>
<td>192</td>
</tr>
<tr>
<td>8.3.4</td>
<td>Extension: Pareto Frontier</td>
<td>192</td>
</tr>
<tr>
<td>8.4</td>
<td>Mappings Involving Capabilities and Adaptations</td>
<td>192</td>
</tr>
<tr>
<td>8.4.1</td>
<td>Approach to Accuracy</td>
<td>193</td>
</tr>
<tr>
<td>8.4.2</td>
<td>Appropriateness of Capability Mapping and Adaptation Suggestions</td>
<td>194</td>
</tr>
<tr>
<td>8.4.3</td>
<td>Extent of Adaptations</td>
<td>198</td>
</tr>
<tr>
<td>8.4.4</td>
<td>Specificity of Capabilities</td>
<td>199</td>
</tr>
<tr>
<td>8.4.5</td>
<td>Cross-device Portability</td>
<td>202</td>
</tr>
<tr>
<td>8.4.6</td>
<td>Relation to Collaborative Situations</td>
<td>204</td>
</tr>
<tr>
<td>8.5</td>
<td>Usage-based Preference Deduction and Compatibility</td>
<td>205</td>
</tr>
<tr>
<td>8.5.1</td>
<td>Compatibility Between Users</td>
<td>209</td>
</tr>
<tr>
<td>8.6</td>
<td>Compatibility with Existing Work</td>
<td>209</td>
</tr>
<tr>
<td>8.6.1</td>
<td>Abstract Interface Generation</td>
<td>209</td>
</tr>
<tr>
<td>8.6.2</td>
<td>Adaptations</td>
<td>209</td>
</tr>
<tr>
<td>8.6.3</td>
<td>Standards and Projects</td>
<td>211</td>
</tr>
<tr>
<td>8.7</td>
<td>Summary</td>
<td>211</td>
</tr>
<tr>
<td>9</td>
<td>Participant-based Test Designs</td>
<td>212</td>
</tr>
<tr>
<td>9.1</td>
<td>Overview</td>
<td>212</td>
</tr>
<tr>
<td>9.1.1</td>
<td>General Challenges and Approach</td>
<td>213</td>
</tr>
<tr>
<td>9.1.2</td>
<td>Participant Groups</td>
<td>214</td>
</tr>
<tr>
<td>9.1.3</td>
<td>Technical Challenges</td>
<td>215</td>
</tr>
<tr>
<td>9.1.4</td>
<td>Ethical Considerations</td>
<td>216</td>
</tr>
<tr>
<td>9.2</td>
<td>Interaction Workflow</td>
<td>217</td>
</tr>
<tr>
<td>9.2.1</td>
<td>Background: Proposed Workflow</td>
<td>217</td>
</tr>
<tr>
<td>9.2.2</td>
<td>Representation of Adaptations</td>
<td>217</td>
</tr>
<tr>
<td>9.2.3</td>
<td>Exploratory Test Scenario</td>
<td>219</td>
</tr>
<tr>
<td>9.2.4</td>
<td>Question Design</td>
<td>221</td>
</tr>
<tr>
<td>9.2.5</td>
<td>Stimuli</td>
<td>221</td>
</tr>
<tr>
<td>9.2.6</td>
<td>Early Lessons Learnt on Methodology</td>
<td>222</td>
</tr>
<tr>
<td>9.3</td>
<td>Longitudinal</td>
<td>223</td>
</tr>
</tbody>
</table>
D Adaptation Classification 271
   D.1 Concrete Adaptation Classification ............................... 271
      D.1.1 Scope of Application ........................................ 271
      D.1.2 Mechanisms .................................................. 272
      D.1.3 Effects on System ............................................ 273
   D.2 Abstract Adaptation Classification ............................... 274
   D.3 Behavioural Classification Revisited .............................. 274
   D.4 Decisions about Adaptations ...................................... 275

E Example Adaptation–Capability Mapping 278

F Publications and Seminars 280
   F.1 Conference Papers .................................................. 280
      F.1.1 Adaptive Accessibility ........................................ 280
      F.1.2 Web and Document Accessibility .............................. 280
      F.1.3 Game Accessibility ............................................. 281
   F.2 Workshop Papers ................................................... 281
      F.2.1 User Modelling ................................................ 281
   F.3 Book Chapters ...................................................... 281
   F.4 Held Workshops and Seminars ..................................... 281
      F.4.1 External ......................................................... 281
      F.4.2 Internal ........................................................ 282
      F.4.3 Workshops for Research Students ............................ 282
# List of Figures

1.1 Relationships between this thesis, Sus-IT and wider research. . . . . 3  
2.1 The Seeheim Model (1983) . . . . . . . . . . . . . . . . . . . . . . 10  
2.2 Example of the personalised menus feature (from [55]). . . . . . . . 14  
2.3 Illustrative spectrum of users. . . . . . . . . . . . . . . . . . . . . . 25  
2.4 (Three-dimensional example of) Capability-space. . . . . . . . . . . 26  
2.5 Several possible relationships between “usability” and “accessibility” . 29  
2.6 Four possibilities for presence and absence of Assistive Technology  
when accessing the web . . . . . . . . . . . . . . . . . . . . . . . . . 42  
2.7 Local and remote collaboration scenarios . . . . . . . . . . . . . . . 54  
2.8 Distributed collaboration scenarios . . . . . . . . . . . . . . . . . . 55  
2.9 Relationships between this thesis, Sus-IT and wider research. . . . . 56  
3.1 Rendering by type (requiring minimal code) in K: default for lists. . 61  
3.2 Rendering by type (requiring minimal code) in K: chart. . . . . . . . 61  
3.3 Example channels. . . . . . . . . . . . . . . . . . . . . . . . . . . . 63  
3.4 A classic Shannon information system. . . . . . . . . . . . . . . . . 64  
3.5 The importance of using the same units in comparisons . . . . . . . 64  
3.6 Frequency response—speakers . . . . . . . . . . . . . . . . . . . . . 65  
3.7 Frequency response—humans . . . . . . . . . . . . . . . . . . . . . . 65  
3.8 Frequency response—environmental limitations . . . . . . . . . . . . 65  
3.9 Conceptual layers of reasoning . . . . . . . . . . . . . . . . . . . . . 70  
3.10 Possible adaptations example . . . . . . . . . . . . . . . . . . . . . 71  
3.11 Adaptation sets decision tree example . . . . . . . . . . . . . . . . 71  
3.12 Architecture of proposed approach—run-time components. . . . . . 73  
3.13 Architecture of a current system with no AT installed. . . . . . . . . 75  
3.14 Current monolithic AT arrangement. . . . . . . . . . . . . . . . . . 75  
3.15 Proposed adaptation-based system. . . . . . . . . . . . . . . . . . . 76  
3.16 Example capability engagement games. . . . . . . . . . . . . . . . . 79  
3.17 Adaptation set selection dialogue box. . . . . . . . . . . . . . . . . 81  
3.18 “Help button” mock-up and Sus-IT software version . . . . . . . . . 81
8.1 How the global indicators are determined ........................................ 187
8.2 Example of orthogonal indicators as axes ........................................ 189
8.3 Mapping processes and data ........................................................... 193
8.4 Adaptation–capability and capability–adaptation cycle from user’s perspective (not including wider preference-related processes) ........................................ 193
8.5 Example extended (from Figure 6.14) mapping from adaptations to capabilities ........................................................... 198
8.6 Four adaptation–capability mapping types ........................................ 200
8.7 Two users’ preference distributions in a given situation ......................... 206

9.1 “Help button” mock-up and Sus-IT software version ............................. 218
9.2 Adaptation set selection dialogue box ............................................... 218
9.3 Architecture of pilot system ............................................................ 220

A.1 Word ’97 (top) and 2003 (bottom) default toolbars .............................. 252
A.2 Two versions of the Word ribbon: 2007 (top) and 2010 (bottom; with improved contrast). From [113]. ........................................................... 252

B.1 The two usage scenario primitives .................................................... 258
B.2 Examples of more complex usage scenarios ...................................... 258
B.3 Further examples of more complex usage scenarios ................................ 258
B.4 An example multiple-environment usage scenario (the dashed lines connect the nodes of interest in the following, detailed diagrams). 259
B.5 Contemporary desktop/portable: composite diagram ............................ 265
B.6 Direct-rendering: composite diagram ................................................ 265
B.7 Local processes: composite diagram ................................................ 266
B.8 Embedded with external Controller: composite diagram ....................... 266
B.9 Embedded with internal Controller: composite diagram ....................... 267

C.1 Layered capability standards .............................................................. 269
C.2 The relationship between the schema(s), standards documents and profiles ........................................................... 270
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Complete initiative-based classification of adaptive systems</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>Comparison of adaptive accessibility features in common OSs (latest versions; including bundled applications)</td>
<td>27</td>
</tr>
<tr>
<td>2.3</td>
<td>Comparison of accessibility features in common operating systems (latest versions; including bundled applications)</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Comparison of approaches discussed.</td>
<td>62</td>
</tr>
<tr>
<td>4.1</td>
<td>Summary of the dimension and temporal nature of static information, streams and control events</td>
<td>92</td>
</tr>
<tr>
<td>4.2</td>
<td>Fundamental modalities, and cognition.</td>
<td>93</td>
</tr>
<tr>
<td>4.3</td>
<td>Fundamental data types (machine perspective)</td>
<td>94</td>
</tr>
<tr>
<td>5.1</td>
<td>Properties of cursors in different UI systems</td>
<td>104</td>
</tr>
<tr>
<td>5.2</td>
<td>Formal properties of cursors.</td>
<td>105</td>
</tr>
<tr>
<td>5.3</td>
<td>Examples of synchronisation properties (output)</td>
<td>109</td>
</tr>
<tr>
<td>5.4</td>
<td>Constraints affecting the collaboration and users.</td>
<td>116</td>
</tr>
<tr>
<td>5.5</td>
<td>Cost factors affecting collaborations and users.</td>
<td>116</td>
</tr>
<tr>
<td>5.6</td>
<td>All possible types of collaboration, based on different cost factor goals.</td>
<td>117</td>
</tr>
<tr>
<td>6.1</td>
<td>Coarse summary of possible adaptation usage and consequences</td>
<td>137</td>
</tr>
<tr>
<td>6.2</td>
<td>Possible consequences.</td>
<td>138</td>
</tr>
<tr>
<td>6.3</td>
<td>Sources of changes to channel capability and requirements.</td>
<td>141</td>
</tr>
<tr>
<td>6.4</td>
<td>Example ranking of abstract adaptations.</td>
<td>147</td>
</tr>
<tr>
<td>6.5</td>
<td>Worked Example of adaptation application</td>
<td>152</td>
</tr>
<tr>
<td>6.6</td>
<td>Compromise in real-world adaptations</td>
<td>152</td>
</tr>
<tr>
<td>6.7</td>
<td>Example adaptations and capabilities mappings</td>
<td>161</td>
</tr>
<tr>
<td>6.8</td>
<td>Example capability implications, based on adaptation usage, during a period of usage (user- and period-specific).</td>
<td>162</td>
</tr>
<tr>
<td>6.9</td>
<td>Scores for adaptations, given normalised weightings.</td>
<td>162</td>
</tr>
<tr>
<td>6.10</td>
<td>Compatibility in T-device layer.</td>
<td>170</td>
</tr>
</tbody>
</table>
6.11 Compatibility in mutation layer. . . . . . . . . . . . . . . . . . . . 171
6.12 Potential conflict scenarios between users. . . . . . . . . . . . . . 172
6.13 Example contextual descriptors. . . . . . . . . . . . . . . . . . . . 176

7.1 Example trade-offs between battery life and capabilities. . . . . . . 181

8.1 Complete Search Performance . . . . . . . . . . . . . . . . . . . . 191
8.2 Seeded Search Performance . . . . . . . . . . . . . . . . . . . . . . 191
8.3 Results of mapping simulations in normal mode. . . . . . . . . . . . 196
8.4 Results of mapping simulations in discovery mode . . . . . . . . . . 197
8.5 Examples of cross-device font sizing. . . . . . . . . . . . . . . . . . 203
8.6 Example T-device viewing distances. . . . . . . . . . . . . . . . . . 203
8.7 Accuities related to T-devices. . . . . . . . . . . . . . . . . . . . . . 204
8.8 Predicted font sizes with acuity 0.013. . . . . . . . . . . . . . . . . 204
8.9 Results of abstract adaptation usage analysis (ranking error and
   indicator scoring factors). . . . . . . . . . . . . . . . . . . . . . . . 207
8.10 The simple indicator-scoring algorithm does not accommodate more
      linear usage profiles. . . . . . . . . . . . . . . . . . . . . . . . . . 208
8.11 User indicator scores and score prediction. . . . . . . . . . . . . . 210

9.1 Workflow trials overview: comparing phases of acceptance testing. . 221

A.1 Comparison of mouse clicks needed to achieve actions in Word 2003
    and Word 2010 . . . . . . . . . . . . . . . . . . . . . . . . . . . . 253

D.1 Adaptation scope-derived names. . . . . . . . . . . . . . . . . . . . 272
D.2 Complete initiative-based classification of adaptive systems . . . . 274
D.3 Detailed abstract adaptation consequences. . . . . . . . . . . . . . 276
D.4 Accept and reject action implications. . . . . . . . . . . . . . . . . 277
Chapter 1
Introduction

This thesis discusses the challenges and opportunities facing the field of accessibility, particularly as computing becomes ubiquitous. It is argued that a new approach is needed that centres around adaptations (specific, atomic changes) to user interfaces and content in order to improve their accessibility for a wider range of people than targeted by present ATs. Further, the approach must take into consideration the capabilities of people at the human level and facilitate collaboration, in planned and ad-hoc environments.

There are two main areas of focus: (1) helping people experiencing minor-to-moderate, transient and potentially-overlapping impairments, as may be brought about by the ageing process and (2) supporting collaboration between people by reasoning about the consequences, from different users’ perspectives, of the adaptations they may require.

A theoretical basis for describing these problems and a reasoning process for the semi-automatic application of adaptations is developed. Impairments caused by the environment in which a device is being used are considered. Adaptations are drawn from other research and industry artefacts. Mechanical testing is carried out on key areas of the reasoning process, demonstrating fitness for purpose.

Several fundamental techniques to extend the reasoning process in order to take temporal factors (such as fluctuating user and device capabilities) into account are broadly described. These are proposed to be feasible, though inherently bring compromises (which are defined) in interaction stability and the needs of different actors (user, device, target level of accessibility).

This technical work forms the basis of the contribution of one work-package of the Sus-IT project, under the NDA programme of research in the UK. Test designs for larger-scale assessment of the system with real-world participants are given. The wider Sus-IT project provides social motivations and informed design decisions for this work and is carrying out longitudinal acceptance testing of the processes developed here.
1.1 Overview of Technical Motivations

A range of different types of interface are used for computer programs and systems and, due to present trends towards ubiquitous, web-based, social and collaborative applications, this diversity is increasing. However, interfaces can present barriers to their use; both “usability” and “accessibility” barriers are discussed, as is their relationship to each other and the fact that accessibility barriers in particular may affect a much larger range of people than most imagine.

Major problems face the expanded use of ATs to overcome accessibility barriers. These include users’ lack of awareness of available sources of assistance as well as mainstream ATs’ current focus on those with moderate-to-severe, often static, impairments. Researchers have developed alternatives: adaptations that focus on specific accessibility challenges, but these are often highly specific and not available to the general public. Adaptations also require infrastructure to support them and their application.

The model of adaptations to overcome barriers fits well with the dynamic diversity [50] of users, devices, applications and the environments in which they are used mandates the need for adaptations to be applied flexibly and semi-automatically. Whilst much work has been carried out on accessibility, ubiquitous applications, adaptations and even some forms of supporting infrastructure for them, little work has been done to determine the viability of a reasoning process to facilitate semi-automatic application of adaptations over time, for a given user, with awareness of the interaction between users, devices, environments and applications. In order to transition from today’s approaches to interface design and accessibility to a more adaptive approach in future, techniques are required to help address current accessibility problems whilst providing a foundation for future systems.

1.2 Contributions

The aim of this work is to develop a reasoning process for adaptive accessibility as described above. As discussed, the work forms part of a funded project which, in turn, is also part of the wider body of research. Figure 1.1 provides a visual description of the contributions of this thesis and Sus-IT.

The contributions of this thesis are as follows. The theory developed seeks to complement and build upon existing work (such as the transformation of abstract interfaces into concrete widgets, for individual users and devices) and, as such, is designed to take a compatible approach.

- Reasoning about accessibility, in terms of Device, User, Environment and
Time (DUET) constraints—thus mapping users with minor-to-moderate impairments to appropriate accessibility adaptations.

- Enabling “good-enough” bootstrapping on new devices or in new situations, based on users’ known capability (and underlying) preferences.

- Operating as passively as possible, allowing users to benefit either from improved adaptability or more active adaptivity, at their choice.

- Supporting accessible ubiquity and collaboration (particularly with respect to output rendering on shared devices).

- Providing a unified method for communicating to the adaptivity system; minimising interaction with ATs and maximising interaction with the application’s interface—both for end-users and, equally, platform vendors and developers.

1.3 Thesis Structure

Chapter 2 is the literature review.

Chapter 3 looks at approaches which may be used to solve the problems posed. This chapter also introduces the pre-requisite theoretical concepts that must be defined before developing the rest of the proposed technique.

Chapter 4 defines the most basic elements of the reasoning process.

Chapter 5 establishes techniques for affording accessible collaborations. Building on the level of capability reasoning developed so far, this allows for a
worked example in which a person with a severe impairment wishes to work with a person without impairment.

Chapter 6 introduces the notion of an adaptation and its effects on information transfer. This allows the capability reasoning process to be refined and developed in order to cope with less clear-cut situations, involving minor-to-moderate impairments.

Chapter 7 discusses some challenges involved in maintaining the effectiveness of the system over time and proposes two main ways in which the work could be extended to facilitate this.

Chapter 8 is a technical analysis of the proposed approach.

Chapter 9 proposes high-level designs for a range of tests of the system that cover different aspects of the reasoning process and how people may interact with and benefit from it. Acceptance testing is to be carried out by Sus-IT.

Chapter 10 presents the conclusions that can be drawn from this work and makes proposals for future extensions of it.

Supplementary material is provided in the appendices, described below. Most of this material exists for one of two reasons, either: (1) to elaborate on how the theoretical techniques developed here are being applied in real-world scenarios or (2) to provide reference information to assist in the reading of the metrics chapters.

Appendix A complements section 2.9’s discussion of the relationship between usability and accessibility by analysing a real-world example (the Microsoft Office “Ribbon”).

Appendix B develops the design of a real-world reasoning system based on the work in this thesis (used as part of the Sus-IT project). It branches off from the work in chapter 3 in a practical direction, whereas the main thesis tackles the theory and reasoning problems.

Appendix C discusses the design of a practical capability classification.

Appendix D gives a theoretical classification of adaptations introduced and developed in chapter 6.

Appendix E contains a sample adaptation–capability mapping used in some of the tests.

Appendix F is a list of publications obtained and seminars given during the preparation of this thesis.
Chapter 2

Literature Review

The context of and motivations for this work are discussed and a number of important terms are defined. Throughout the discussion, various key goals for any computer system intending to be both usable and accessible (defined in the text) are highlighted. These goals—and the range of users that they would benefit—are discussed in the context of the literature and relevant artefacts from industry. Finally, the goals identified are summarised, those being addressed by Sus-IT are listed and the subset of the goals on which the rest of this thesis focuses are highlighted.

The early sections of this chapter define some fundamental concepts on which this work is based and introduce the literature surrounding each of them. Section 2.5 marks the beginning of a discussion on the requirement for improved computer accessibility and current industrial endeavours. From section 2.12 a number of areas of research, some introduced earlier, are drawn on to highlight several potential solutions to the problems raised—and the limitations that have so far prevented their mainstream adoption. The final sections (2.14 onwards) discuss the key remaining problems and lists those to be addressed by the remainder of this thesis.


2.1 Defining Usability

For the purposes of this work, the contemporary definition is used: usability is “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” [63]. This definition is accepted by others [65, 87]. Usability is concerned with the time needed to carry out tasks, the number of steps involved, the overall efficiency of the interface design (e.g. travel time between related commands in the system)
and how the user feels about the usage of a system—a highly subjective matter. Nielsen describes the five main dimensions of usability as: learnability, efficiency, memorability, errors (low error rate; easy recovery) and satisfaction [84].

This work is primarily focussed not on usability but accessibility (the tacit definition will suffice until this is defined from the perspective of this thesis in section 2.5), however there are notable relationships between the fields (discussed in section 2.9) and the relevant literature is discussed throughout the rest of this chapter. A definition that links usability and accessibility is developed in subsection 2.9.4.

2.2 Adaptive Interfaces

Dieterich et. al. carried out a detailed survey to classify research on adaptive user interfaces [28]. Taxonomies were developed to cover the nature of adaptive systems in general and the range of techniques and architectures used for individual components in such systems. Two very important classifications (also used by later work, such as investigations into the potential of configuration agents, acting on behalf of users, to improve accessibility [110]) will be discussed here.

2.2.1 Behavioural Classification

The fundamental stages involved in the process of adaptation, considered from the user’s point-of-view, were highlighted: initiative (the stage at which the need for adaptation is predicted, detected or requested), proposal (where candidate adaptations are offered), decision (one adaptation is selected) and execution [28, fig. 2]. Furthermore, the systems are then classified on the basis of which party—user or system—is responsible for each stage. Table 2.1 is a full expansion of this classification (of 16 possible systems). The classification presented then leads on to the identification of four typical types of adaptive system (i.e. where the system proposes the adaptation) and two types of user-proposed adaptations, as follows.

**Self-adaptation (SI-SP-SD-SE)** is where the computer initiates and carries out all other stages of adaptation. This typically requires deep domain knowledge—a comprehensive task model (defined shortly)—and a reliable user model, as the computer would have to determine the user’s intentions and plan an appropriate adaptation itself.

The reviewers state that this technique is most appropriate for adaptations that reflect the needs of the application—at the time of writing examples of this included context-sensitive help and Graphical User Interface (GUI)-building tools, but today, in the context of ubiquitous computing, this could
Table 2.1: Complete initiative-based classification of adaptive systems, based on Dieterich et. al.’s study [28].

<table>
<thead>
<tr>
<th>H</th>
<th>Name</th>
<th>I</th>
<th>P</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Self-adaptation</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>User-initiated self-adaptation</td>
<td>U</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>Computer-aided adaptation</td>
<td>U</td>
<td>S</td>
<td>U</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>User-controlled self-adaptation</td>
<td>S</td>
<td>S</td>
<td>U</td>
<td>S</td>
</tr>
<tr>
<td></td>
<td>Adaptation</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>System-initiated adaptation</td>
<td>S</td>
<td>U</td>
<td>U</td>
<td>–</td>
</tr>
</tbody>
</table>

○ System-proposed, user-executed (*unreasonable*) – S U U

○ User-proposed, system-decided (*not used*) – U S S

○ System-decided, user-executed (*unreasonable*) – – S U

Notes: Emphasised names/characteristics come directly from the review. There are 16 possible systems as there are four stages of adaptation, each of which being carried out by (the user) XOR (the system). The table is grouped into blocks of four.

Fields: H (highlighted as interesting in paper): ● = yes; ○ = no. Stages: I = Initiative; P = Proposal; D = Decision; E = Execution. Actors: U = User; S = System; – = either (therefore accounting for two combinations).

also include reaction to available network connectivity methods (e.g. Wireless LAN, cellular or lack of connectivity) or the location of the device running the application.

User-initiated self-adaptation (UI-SP-SD-SE) is where the computer carries out all stages after adaptation is specifically requested by the user.

Computer-aided adaptation (UI-SP-UD-SE) requires the user to initiate adaptation and then select from a number of possible adaptations proposed by the computer.

User-controlled self-adaptation (SI-SP-UD-SE) allows the user to choose from a set of proposed adaptations after the computer initiates the process.

Adaptation (UI-UP-UD-UE) is the totally manual form of self-adaptation and, as the review suggests, encompasses situations in which the user changes system settings manually. An example may be changing the desktop background in a Window, Icon, Menu, Pointing device (WIMP) GUI.

System-initiated adaptation (SI-UP-UD-UE) occurs in cases where the system determines it may be useful for the user to make some sort of customisation.
One other key insight that is offered by this review is definition of a two-dimensional classification of the different types of adaptive system that is based on the type of “intelligence” that is required of the system. The two types of intelligence are, essentially: (a) content analysis and plan recognition and (b) proposal creation and evaluation. The former is required for the more system-initiated adaptation scenarios and the latter for the more system-guided adaptation scenarios [28, fig. 3].

It is observed by Dieterich et. al. that systems using the computer-aided adaptation approach seemed the most successful and that projects using only self-adaptation generally failed to prove user satisfaction and thus were generally not worth the implementation cost [28, sec. 6]. This is most likely down to the more hefty requirements for the computer to display apparent intelligence as it assumes responsibility for more stages of the adaptation process.

Finally: “adaptive” systems were contrasted to contemporary “adaptable” systems, which were classified in a different manner, in which “adaptable” systems emphasised user responsibility for more of the stages above, whereas “adaptive” systems gave more of the responsibility to the system.

2.2.2 Architecture

The approach of abstracting user interface management was advocated by a number of researchers. The term “User Interface Management System (UIMS)” was used to talk about the design-time and run-time tools that performed this task [59] (in much the same way that a Database Management System (DBMS) manages the structure, access and maintenance of a database).

Dieterich et. al.’s study observes that there are some standard components involved in all adaptive user interfaces, though the arrangement of these seems to vary considerably across projects—and sometimes is supplemented by further components. The most widely-used components identified at the time of that study are still valid for the contemporary literature (highlighted as appropriate below). It should be noted that not all systems incorporated all of the distinct models listed below. Further, the models may not be implemented in a specific place or process in a system; some systems are able to infer the equivalent data from other sources. The standard popular components are as follows.

**User Models** contain data that have either been learnt or predicted about the user—their capabilities, preferences and usage patterns. An initial model may be populated with “standard” data or with estimates based on a calibration session with a particular user [44].
In domain-specific projects, the user model may contain far more data such as preferred learning styles [104].

**Application/Task Models** define the commands provided by a given application, including required or optional parameters and, in some systems, pre- and post-conditions for such commands [105]. This information gives the UIMS the ability to automatically determine key properties of the interface—for example: if a command’s pre-conditions have not been met, it should be disabled in the interface (this could be achieved by “greying out” or hiding the command, as appropriate).

It is important to note that this knowledge can be used by interface-building tools to make the process of interface design more automatic and is also a key requirement of abstract user interface systems [79], which will be introduced in section 2.4.

**Dialogue Models** were originally a central point in adaptive systems because the main form of interaction with the computer was dialogue-based—i.e. the flow of interaction was conversation-like and the computer dictated the range of valid actions the user could choose at any point. Early dialogue models were based on bespoke specification languages that expressed the input and output requirements of the application; the UIMS would then generate appropriate text-based dialogues for the user to interact with. This approach was used in systems such as COUSIN [59] and Open Dialog [60].

Over time, and with the rise of “direct-manipulation” interfaces, the prominence of the dialogue model has decreased [79]. In some cases, this term refers to a purely conceptual part of the system that is not implemented as a specific code module or process. In the case of UIDE, this conceptual component infers its data from the application model and ensures the correct ordering of input and output is maintained [105].

Various different architectural approaches to implementing a UIMS were proposed; separable user interfaces [77] being one example.

Several architectural models for adaptive systems were developed; the first being the Seeheim model [88] (see Figure 2.1), which was followed by more advanced models that were designed to take into account user preferences so that interfaces could be more tailored to them—e.g. the Knowledge-Based Front-End (KBFE) model, which was based on the Seeheim model. It is important to note that at this stage these preferences did not include issues of accessibility, which will be introduced in section 2.5—however, critics of the Seeheim and related models assert that the user’s experience and abilities should be taken into account [90].
Figure 2.1: The Seeheim Model (1983). Applications are separated into distinct layers/components. Rapid semantic feedback can be passed “back” to other layers.

The other architectural trend in adaptive user interfaces is that as more stages of the adaptation process are controlled by the computer, the more domain-specific knowledge, user models and usage patterns data are required. This is certainly true of later domain-specific adaptive projects, such as AVANTI [104].

2.2.3 Scope of Adaptations

Early adaptive systems were developed at a time just before the industry standardisation on specific WIMP interfaces and widget sets. Developers of applications at that time faced several large challenges such as: which GUI platforms and widget toolkits they could target—each requiring a distinct and large volume of interface code—and the management of the interface logic (housekeeping tasks such as creating and destroying interface elements and keeping track of whether widgets should be enabled or disabled).

Due to this, primary goals of many early adaptive systems were either semi-automating the process of designing and composing the interface elements [78] or, further, generating interfaces that managed their own behaviour at run-time based on the application’s internal structure and semantics [37]. After the widespread commercial adoption of WIMP-class GUIs, the need for this abstraction waned [79]. The recent revival of abstract user interface research is discussed in section 2.4.

From the user’s perspective, portability-focused systems may not even have seemed adaptive, as the interface remained fairly static. However the user’s actions were actually separated from the interface by a layer of models that intercepted the input, ascertained the user’s intention and updated the interface accordingly (i.e. within the conventions of the Operating System (OS) in use). Later iterations of such systems were able to add more user-visible adaptive features, such as guidance on how to carry out certain commands offered by the system, based on the application model [105], taking them closer to being “intelligent user interfaces”.

Other, more esoteric, adaptation techniques were sometimes used, such as switching to a different style of interface that would better suit the user’s style of working—e.g. from WIMP to Command-Line Interface (CLI) [17]. Such a change
was justified by surmising that the user’s spatial awareness and memory abilities may cause them to be more adept at using one type of interface than the other. Another system offloaded the user’s obligation to take part in the stages of adaptation described above in line with their stress levels.

During the course of the 1990s and 2000s, modelling and knowledge-management techniques have been refined, which has resulted in some successful domain-specific adaptive systems; particular areas of success being online shopping and other recommender systems, electronic learning tools and tourism [104]—the AVANTI project, which is also notable for focusing on the needs of users with “recognised disabilities” (impairments that the general public would consider disabilities; the notion of disability will be discussed in more detail in section 2.8), as well as adapting to general interests. Computer games have benefitted from a combination of increased available processing power and the use of Artificial Intelligence (AI) and graph-searching techniques to provide realistic opponents for gamers that are capable of adapting to the players’ skill level. Research has been undertaken on adaptive interfaces with a focus on the needs of people with disabilities [92, 40] and will be discussed in more depth from section 2.5 onwards.

Finally: content, as well as interfaces may be adapted. Examples include the transcoding of multimedia data held on a server [13] to the adaptations of web pages [91].

2.3 Critique of Adaptive User Interfaces

Adaptive user interfaces have not been without criticism—both from academia and as a result of commercial endeavours that have not been well received by end-users. This section details some key objections and challenges that adaptive interfaces face.

Specificity—flexibility trade-off. Adaptive interfaces that are highly-domain specific may be able to afford the user some very useful adaptations. However, those that seek to be more generic may fail to do so, as the problem becomes computationally intractable and, thus, the task model becomes relatively more rudimentary. Myers et. al. refer to a problem of threshold and ceiling—the ideal systems being low-threshold and high-ceiling, which is very hard to achieve generally (though some techniques for allowing the user to discover more advanced features of an interface, such as the “trapdoor” method, hold some promise) [79].

---

1Such as Epic Games’ “Unreal Tournament” originally-released in 1999; http://www.unrealtournament.com/
People do not like change. In the past, some attempts at adaptive user interfaces have met with much resistance from the general public, due to the fact that the interface changed unpredictably (see subsection 2.3.1). It is also the case that developers do not react well to change; changing platforms often brings the requirement to learn new technologies and skills, as well as the associated expenses. However, there are very successful techniques such as application/programming-language Virtual Machines (VMs) that can mitigate these effects to some extent.\(^2\)

The changes that designers and developers appear to be most averse to are those introduced by “intelligent” technologies such as UIMSs and constraint-driven User Interface (UI) renderers, for the same reason that users are against fluctuating UIs—unpredictability [79, sec. 2.3].

Use users, not models. There have been notable and justifiable objections to the notion of using models and guidelines to test systems, rather than users. This can result in lower awareness of the design issues with developers [86]. Objections like this are noted—especially to the apparent abstraction inversion that is occupying the limited power of a computer with the intractable task of simulating the human user. Such a generic simulation should be unnecessary for an adaptation system that targets individuals experiencing minor-to-moderate impairments, and who in some cases can communicate to the computer that an adaptation is required, specify the type of adaptation required (e.g. zooming into a wordprocessor document indicates that the user is having difficulty reading the text), or give feedback—acceptance or rejection—regarding adaptations that were system-initiated.

User model inaccuracy. The accuracy problems with generic, simulating, user models could be great. However, systems that develop statistical models based on the actions of a given user over time, at least in theory, should provide much more accurate guidance as to what the user may deem appropriate in a given situation. Effectively, such a model turns the user into the expert part of an expert system.

One size does not fit all. Of course there are many types of user and disability—and, consequently, user models (such as those listed in subsection 2.11.7). This even applies to models within a particular type of disability, such as motor control [70]. Even when a standard modelling approach is adopted, such as the Model Human Processor, there can be many different concrete

\(^2\)VM found commercial use in the interactive fiction games of Infocom long before Java and scripting languages became popular: \url{http://en.wikipedia.org/wiki/Z-machine}
implementations, to be used in different circumstances and for different dis-
abilities; e.g. [20].

**Bootstrapping** is the process of arriving at a reasonable starting profile and set
of adaptations for a given user. It represents a key hurdle for any adaptive
system, as the system ideally needs time to learn how to be effective for
a given user. The many issues surrounding the bootstrapping process—
from reliability to security—have been raised in other work [91, 110, 40] and
clearly need further investigation.

**Portability** of user profiles and other pertinent data is paramount—users must
be able to interact with any instance of an adaptive system as if it were their
own.

### 2.3.1 Case Study: A Commercial Adaptive Interface

A notable commercial adaptive interface was presented by Microsoft’s “personal-
ised menus” feature [55]. The idea was to reduce the complexity of the interface
of Microsoft’s Office suite of applications; menus that actually contained a lot of
commands would display only the most frequently-used ones to the user, keeping
the UI simple. The user could request to see the full list of commands, if desired,
and could then pick one of the previously-hidden commands from the menu, as
shown in Figure 2.2. That command would stay visible in the menu, so that the
user did not have to expand the entire menu to find it again in future. Commands
that had not been used for some time may be hidden from the menu to avoid caus-
ing visual clutter. This seems to be a reasonable idea, but it has several problems,
as follows.

**Unpredictability.** When the user first uses a command that was not visible by
default in the menu, the items after that command in the menu are moved
down by one. Given that users become familiar with where each desired
command may be found (using motor memory [41]), when the menu order
shifts it is necessary to consciously re-scan the list for the new position of
the command (or, worse, if the user does not notice this change an incorrect
command will be activated). This can cause disorientation and frustration
for users.

**Lack of portability.** The system was not designed to handle the case where users
move between computers. In these cases, the menu system would not reflect
a user’s usage of the system and the problem of unpredictability would arise
again during usage of the new system.
As a result of these shortcomings, the system was widely unpopular and has since been deactivated by default [55]. However, as is demonstrated by Gajos, users were more productive—and happier—when the adaptive list of frequently-used commands was confined to a particular area of the overall UI, as this kept the rest of the interface stable and predictable [41] (a technique known as “split menus”).

2.3.2 Adaptability

Related to the concept of adaptivity is that of adaptability. This concerns a system that has been designed to be highly customisable, but there are no active adaptations—the user is responsible for making all customisations [33, pp. 2]. Adaptable systems have all of the benefits, particularly in accessibility terms, but do not introduce any often-controversial autonomous or even semi-autonomous adaptations—in fact, even when they do not improve user performance, users can perceive an improvement, possibly because the feeling of having more control is considered even more important [33, pp. 7].

In fact the processes developed in this thesis are in some ways more adaptable than adaptive, because they aim to reflect the user’s will with the goal of ensuring that devices and collaborations are accessible “out of the box”.

Figure 2.2: Example of the personalised menus feature (from [55]).
2.4 Ubiquitous Computing and Abstract Interfaces

As discussed in subsection 2.2.3, early adaptive interfaces were aimed at portability across multiple desktop computing platforms. These later developed into “intelligent user interfaces” that provided value-added features such as automatically-generated context-sensitive help. However, the need for adaptation for portability declined due to the advance of WIMP interfaces and the thinning of the market for alternative OSs.

2.4.1 The “Everyday Computing” Vision

Since the mid-1990s, the anticipated diversification of computing devices (significantly in terms of scale and mobility) has occurred, resulting in Personal Digital Assistants (PDAs), “smartphones”, large interactive display screens and, very recently, netbook computers. However, the multitude of form-factors was seen as secondary to the applications they could support, summarised by Abowd et. al. [1] as facilitating: (a) more natural recognition-based input methods such as writing, gestures and speech; (b) greater awareness of context including location, which may be used to provide augmented reality services (which are becoming popular on smartphones\(^3\) and are exemplified by the audiogame “Demor” [116]) and (c) the continuous capture and archival of users’ life events and their later retrieval.

With the advent of multi-touch devices with post-WIMP GUI interfaces specifically designed to embrace these more natural means of direct manipulation, the ubiquitous computing vision is progressing into being mainstream. In fact, such devices could be seen as the culmination of Jeff Raskin’s vision of “Information Appliances” popularised by Norman [85]—devices that are dedicated to one specific information-related task (e.g. retrieval, communication or entertainment) and, thus, are argued to be significantly more straightforward to use than a “traditional” computer. Modern multi-touch devices such as the iPhone are able to reconfigure themselves as one of a multitude of specific appliances on demand—performing functions that would have previously been carried out by physically distinct devices, yet without burdening the user with the cognitive overhead of managing several concurrently-running and possibly overlapping applications, as is usually the case with WIMP UIs.

This section summarises the aspects of pervasive computing most relevant to the reasoning techniques developed in this work, which are are more output-focused and, due to the motivation of improving access to some legacy systems

---

\(^3\) e.g. [http://mashable.com/2009/12/05/augmented-reality-iphone/](http://mashable.com/2009/12/05/augmented-reality-iphone/)
as well as new ones, focused on solving more medium-term problems. However, a number of concepts from the pervasive computing vision, such as collaborative and distributed applications are considered—and the techniques developed are low-level and thus are intended to make contributions to future interaction paradigms in due course.

2.4.2 Renewed Need for Abstraction

Due to the proliferation of new form-factors for computing hardware, the need for portability arose once again [79, 1], the motivations being two-fold, as follows.

Content portability was required to avoid massive expenditure on re-writing or transforming content for mobile devices. Fracturing of mobile web standards, for example, resulted in various subsets of HTML being used on various devices of widely different capabilities.

Since the early 2000s, the trend has been towards mobile devices becoming capable of rendering full HTML and the content provider using technologies such as eXtensible Stylesheet Language (XSL) filters being applied to the “standard” version of sites to enable them to render appropriately on a mobile device. However, problems of screen size and information overload still remain.

Adaptations to content by transcoding on servers has been investigated as a way to mitigate bandwidth and hardware constraints on the part of the device as well as any sensory or other impairments the user may have [13]. Transcoding in respect of bandwidth constraints appears to be standard practice, though not for accessibility reasons.

Interface portability techniques such as UIMSs once offered the ability to design an interface once and deploy it across multiple devices. However, it has largely been accepted that attempting to render the same interface on desktop and mobile devices (a form-factor that did not previously exist) is going to severely impair the user experience on at least one device. Further, some developers consider it prohibitively expensive to develop many user interfaces to their application (though this has happened in some cases, particularly where large markets have been foreseen).

There is research interest in the specification of interfaces at a higher level of abstraction (see below), as this may allow deployment on multiple devices,

\footnote{This technique was used by TESCO to provide an “accessible” (defined in section 2.5) version of their on-line shop to customers with disabilities.}
where each device would render interface elements in an appropriate manner and skip or substitute those it cannot support (possibly falling back to simpler elements in its place) [36, 94]. Other alternatives include distributing the interface for an application across devices.

A number of research and development projects have created abstract interface specification libraries, such as those discussed by Trewin in the context of supporting people with disabilities [111]—User Interface Markup Language (UIML), eXtensible Interface Markup Language (XIML), W3C’s XForms and Universal Remote Console (URC). Many other abstract interface standards exist and are discussed by other literature, including: Mozilla’s XML User interface Language (XUL); Microsoft’s eXtensible Application Markup Language (XAML); Style-Based Markup Language (SBML) [47] and IBM’s Abstract User Interface Markup Language (AUIML) (retired).

### 2.4.3 Generality

Other definitions are particularly concerned with people with disabilities. A key aspect of any truly abstract specification language would be to describe the semantics of interaction without referencing specific concrete implementations, e.g. WIMP GUI widgets. Several specific “interaction styles” have been identified, such as form-filling and back-and-forth dialogue and are expressible in SBML [48]. Some, but not all, of the standards listed above support these—for example XUL and XAML use WIMP-specific terminology as they were envisioned only for creating such graphical applications. Also, SBML is couched in GUI-centric terms such as “button” and “window”. Some of the early dialogue-based adaptive systems, such as COUSIN, define the UI in terms of input and output requirements, but also use terms such as “pushbutton” to describe the data types [59]. Although it is not possible to design a completely generic system, it is important to avoid bias towards contemporary interaction styles when trying to create a system that is agnostic to interaction style.

Some projects consider the challenge of making control of a multitude of ubiquitous devices easier; e.g. Personal Universal Controller (PUC) [82], or URC [126]. This is a mobile device running software that generates interfaces at run-time for any particular household appliance. The functions of the appliance are described by an abstract specification and both GUI and spoken interfaces can be generated. This solves one of the problems that the standardisation on WIMP interfaces solved—it gives the user a standard means for service discovery and interaction. It also opens the door to easing access for people who cannot access traditional types of GUI. It does not claim to be useful to people with disabilities—in fact Total
CHAPTER 2. LITERATURE REVIEW

Access System [96] is focused on this—but the ability to generate speech-based interfaces raises the possibility that it may be with a small amount of additional effort. As with the abstract UI specification languages above.

However, it is asserted that an accessible UI to a blind user should not necessarily be a Text-To-Speech (TTS) representation of a GUI (which is, in turn, a representation of the underlying model)—it should not require the user to be aware of GUI terminology; it should appear to be the interface to the system, as opposed to an “accessibility layer” on top of another interface. The primary reason for this assertion is that users should not be expected to think in terms of more than one mental model simultaneously in order to interact with a system, particularly when one of those mental models requires use of a sense the user may not possess.

2.4.4 Criticisms

Some of the challenges facing ubiquitous computing are beyond the scope of this work, such as: input recognition errors; and the development of more sophisticated wearable computing [1].

Other criticisms may include the fact that it is not currently feasible to use abstract interface techniques as they still yield inferior results to human designers. However, the exclusive use of such techniques is not necessarily advocated even by researchers developing them (including Gajos, who recognises the importance of balance between human design input and mechanical assistance [40, ch. 7]). The development of such techniques is of key importance because of the flexibility it instills in applications, which is of particular value to users with impairments brought about by extreme environments or perhaps recognised disabilities.

2.4.5 Collaboration

In a world where computational devices are ubiquitous, it is inevitable that they will be used to collaborate with other people. This began some time ago on the desktop—via networked shared spaces, computer games and social websites—and has become popular on mobile devices, via telephone, SMS and video services, as well as Personal Area Network (PAN)-enabled applications [25].

A detailed ethnographic study of a group of older web users has highlighted the use-case in which a group of users share one device to complete a task collaboratively [95]. This, in turn, highlights the need for assistive technologies to be aware of such a scenario and react appropriately to it. (It should be noted that the main barriers discovered by the cited research were of the “mismatched mental
models” and content-related cognitive types—however the focus of this thesis is on the perception-related accessibility aspects.)

There is a large body of research on topics such as Computer-Supported Cooperative Work (CSCW) [52], including work on the challenges of service discovery in a ubiquitous computing context [94] and the challenges of supporting devices with different capabilities [36]. The focus of this thesis, as will be discussed in more detail later, is more on providing access to a given collaboration rather than on the mechanics of the collaboration itself; relevant work from this area of research will, therefore, be drawn upon in future chapters.

2.4.6 Accessible Collaboration Case Study

One exemplary case in which collaboration has been afforded between people with very different capabilities is that of Apple’s “VoiceOver” screenreader software VoiceOver is a kind of AT that allows people with severe sight impairments to use Apple’s Macintosh computers. VoiceOver is important for two reasons, as follows.

**Bundling.** VoiceOver comes as standard as part of Mac OS X, so it is always available (as is the screen-magnification functionality). This means users can always be confident that assistance is available. It is also always activated in a standard way.

**Interaction inclusion.** Fully-sighted users often find watching a blind person interact with a computer confusing, due to the use of (often fast-paced) TTS or Braille output. VoiceOver combats this by strongly emphasising the currently-focused GUI item, so that the sighted user can instantly see where the blind user is in the interface.

Both of these reasons in combination were revolutionary at the time of the product’s introduction (April 2005) and, unfortunately, appear to remain so, as there are no other provisions for accessible collaboration that are universally available on other computing platforms.

Despite the clear advances made by VoiceOver, however, the reader may have noticed that in subsection 2.4.3 the notion of forcing people to be aware of an interface paradigm that they are accessing only indirectly and therefore incidentally (e.g. a GUI, via TTS only) was considered detrimental, yet here, the VoiceOver approach—building in a layer above the GUI to enable access—is being praised. This will be revisited in subsection 2.11.2, after some important definitions have been made.

**Goal 1 (Collaboration).** Support collaboration in an accessible way.

---

5http://en.wikipedia.org/wiki/Mac_OS_X_Tiger


2.5 Defining Accessibility

Paraphrasing the definition of usability from section 2.1, if a system was absolutely accessible, then its usability would be “the extent to which a product can be used by all users to achieve specified goals with effectiveness, efficiency and satisfaction in any context of use”—the emphasis being placed on clauses which have been changed from the ISO definition of usability. The changes reflect the notion that systems may be used by a wider range of people and in a wider range of contexts than were initially envisaged. Therefore, for the purposes of this work, the accessibility of a system is the extent to which the changes in user characteristics and usage scenarios are supported. This section also gives some contemporary definitions of accessibility; later sections contrast this to the definition used by this thesis.

Practically, the proposed definition of accessibility means that a product or service should meet the highest-level guidelines from the W3C [120, sec. “Introduction”], which stipulate that it should exhibit the following characteristics.

**Perceivable** by as many users as possible in as many contexts as possible.

**Operable** as above.

**Understandable** a vital property, but largely out-of-scope for this work.

**Robust** which seems to align with Nielsen’s “errors” dimension of usability. However, as robustness should be considered for all elements of a system, including user-facing components, it is not considered accessibility-specific by this work.

These four principles are known as the “POUR” principles—though it is asserted that any system must also be reachable (economically, technologically and culturally) in addition. These goals seem perhaps more fundamental than the dimensions of usability given in section 2.1 and must be satisfied in order for any user to access a given system. At the highest level, this definition of accessibility broadly reflects that currently widely-used in the Information and Communication Technology (ICT) industry and academic research—but there is no implication here that it is mainly a matter related to catering for users with recognised disabilities. A definition that links usability and accessibility is developed in subsection 2.9.4.

It is accepted, of course, that no system may be usable to everyone, in every usage context. The goal of accessibility provision, therefore, must be to maximise the chances of this for any given user(s) and scenario(s). Much of the rest of this
chapter argues that although current methods employed to achieve accessibility have made great advancements, a new approach is needed to enable mainstream adoption.

2.5.1 Broad Definitions

The word “accessibility” is used in both everyday language and in specific fields and its broad definitions vary considerably: everyday usage implies the qualities of being at hand, approachable, reachable when needed [65];

6 financial accessibility is the affordability of a product or service for a given person; the accessibility of place, with respect to transport, is a function of how connected it is to other places.

2.5.2 Public Perception

As a barometer of how the notion of accessibility is currently held by the public, consider its article on Wikipedia. The article points out that that accessibility is not necessarily related to people with disabilities, but the majority of the article is concerned with accessibility provisions in different fields for those with recognised disabilities. The article relates accessibility strongly to universal design (see section 2.7), which aims to apply to everyone—not just those with recognised disabilities. It is also introduces the “indirect access” approach, embodied by the use of ATs for a disabled person (discussed in section 2.10).

These two approaches to improving accessibility—universal design and ATs—are currently the most widely-used methods but, as will be discussed, they have limits and may not be the most appropriate way to bring accessibility provision into future mainstream ICT systems.

2.5.3 Accessibility in terms of Barriers

Whereas usability may be concerned with how quickly a user can accomplish specified tasks, the level of effort that is required or even how using a system makes a user feel, it may be helpful to consider accessibility in terms of the barriers a user might face in gaining access to (perceiving and operating) a system. Examples of situations which could present barriers include the following.

6http://www.google.com/search?q=define:accessibility
7http://en.wikipedia.org/wiki/Accessibility#Transportation
8at least those that have some interest in the subject
Reading text on a computer screen. A barrier may be presented by the size of the text.

A set of stairs. A barrier may be presented by the number or gradient of the steps.

Reading information expressed in a given language. A barrier may be presented by the requirement to know the language.

From this viewpoint, accessibility could be seen as a gap between the combined requirements of \{the environment, device, product or service\} in use and the resources or abilities that the combination of \{user, device and environment\} are able to offer. In some cases it may be possible to precisely quantify this gap, as the examples below illustrate.

- The gap through which a wheelchair user wishes to travel may be 3cm narrower than their chair.
- A user may be unable to interact with a GUI system using a mouse.
- Ambient noise levels may reduce a user’s ability to perceive speech.

Likewise, in some situations the accessibility problems may be evident but not well understood—e.g. a user may be unable to understand some aspects of the interface due to cognitive problems such as dementia (though, again, this is out-of-scope for the present work).

Naturally it may not always be possible (particularly with physical-world situations) to compensate for accessibility barriers. As is discussed in section 2.10, software and hardware has been developed in an attempt to counteract these problems in computer systems—therefore bridging the accessibility gap.

The differences between accessibility and usability will be further contrasted by a real-world example in section 2.9 and Appendix A.

2.6 Defining (Computer Interface) Accessibility Barriers

A person may experience problems interacting with a computer system for various reasons. The nature of accessibility barriers may be classified as problems with the following.

Environmental constraints such as inappropriate lighting types, levels or high background noise.
**Device constraints** such as a small screen\(^{10}\) or inability to output in a modality that the user requires (e.g. Braille).

**Functional, sensory or cognitive impairment** on behalf of the user, possibly brought about by disability or the ageing process [71, 91, sec. 3].

**Learning style mismatch** between that of the user and that for which the system was designed. Numerous models for “learning style” exist [58], but the most relevant to this work is Fleming’s visual/auditory/kinaesthetic model, which has clear links to certain types of content, and certain ways of rendering that content, being more appropriate for a person than others.

A stark example of this is the early set-top media boxes; they contained options such as “Videos”, “Pictures” and “Music” whereas the user would have preferred “My holiday in Blackpool 2010”. The learning styles of the developers (highly structured and type-based) and users (socially-based) did not match.

**Temporal constraints** such as: (a) “information overload”—a phenomenon that may still easily occur in the absence of any recognised cognitive or other disability, or (b) a lack of time for the user to react accordingly (which may, again, be caused by disability).

### 2.6.1 Scope of these Barriers

Any one of the above conditions may present an accessibility barrier to potential users of a system. It is quite possible that some users will experience one or more such problems simultaneously—a prime example of this that of older people, who may develop multiple minor disabilities [71, 51, 91].

It is important to consider that these difficulties may be temporary, such as those caused by environmental changes (e.g. moving from inside a building to outside and back), transient medical conditions (such as broken limbs) or fluctuations in a long-term condition, such as vision impairment.

From the above, it is evident that any accessible system must meet the following criteria.

**Goal 2** (Accessibility barrier causes). *Take into account that accessibility problems may be caused by a wide range of factors, including functional, sensory or cognitive impairments of the user, or limitations imposed by the device in use, the environment or situation (context) of an interaction.*

---

\(^{10}\)Note that “small screen” does not need to be qualified—to any human using an ICT it would be obvious what constituted a small or a large screen, as we all share the same basic anatomy and capabilities. This point will be expanded upon in the following chapter.
Goal 3 (Simultaneous accessibility barriers). Recognise that zero, one or more functional problems may exist at a given time.

Goal 4 (Temporal flexibility). Accept that accessibility problems may—in fact are very likely to—vary over time. Further, there may be general trends in capability change, over multiple timescales.

A system meeting both Goals 2 and 4 would consider the complete set of data needed to reason about the accessibility of a particular interaction, which will be referred to as the Device, User, Environment, Time and Situation (DUETS) constraints.

2.7 Universal Design

One approach that has developed in an attempt to work around these various difficulties is that of “universal design”, sometimes expressed as “design for all”. Keates and Clarkson detail many examples of physical-world products that, whilst originally designed with disabled people in mind, have gone on to become successful in the wider market, due to being suitable for use by a wide range of people. These include “big button” telephones, OXO kitchen utensils and kerb cuts [24, pp. 11–15]. Similar widely-useful product features include the bumps on certain keys in keyboards and keypads.

Other areas in which universal design has been employed include: using ramps (as opposed to stairs) for building access; providing adjacent sets of water fountains and cash machines at different heights and the addition of audible and tactile feedback for pedestrians at traffic lights.

Due to the static nature of most physical artefacts, they cannot always be designed in such a way that renders them suitable for everyone; many people have conflicting access needs, e.g. short and very tall people, those with light-sensitivity and those requiring bright or high-contrast displays. Efforts to design products and environments with the aim of including as many different people as possible should, of course, be encouraged, but it must be acknowledged that the “average user” almost always does not exist [69].

The relative inflexibility of physical objects is in stark contrast to the electronic world, where all rendered output could, potentially, be adjusted in real time—however, as will be discussed, this opportunity is currently only rarely taken; there is still a need for specialist ATs to enable those with specific needs to access electronic systems, as many such systems attempt to mirror their real-world counterparts (electronic paper systems being a prime example).
CHAPTER 2. LITERATURE REVIEW

2.8 A Spectrum of Capabilities

Section 2.6 discussed that many people are likely to experience accessibility barriers of some sort over time. What is not clear is to what extent these accessibility barriers will affect them. Depending on the cause of the accessibility barrier, there may be any number of adaptations that can be made, some by the user themselves without even realising it (e.g. turning the volume of a television up; putting on reading glasses; moving to a better vantage point). Sometimes, however, it may not be practical to employ these methods. A simple example of this would be someone at a railway station who is struggling to read the departure board due to glare. It may be that the visual acuity of some people is sufficient to read the display even in poor lighting. Failing this, the simplest solution for most people would be to change their position or find a different board to read. However, if the person concerned is vision-impaired or has a mobility difficulty (or both) then it may not be possible for them to implement a method to overcome the barrier.

The influence of accessibility barriers and a person’s ability to overcome them will depend on their position on a spectrum of capability (Figure 2.3). In fact, as there are many possible capabilities a person may have, there will be spectra in many dimensions; a given person’s capabilities may be thought of as a point in capability-space (Figure 2.4).

Goal 5 (Compensating for accessibility barriers). Recognise that potential accessibility barriers only become actual barriers when the user cannot find a way to compensate for them. A theoretical ideal computer system would, therefore, be capable of suggesting possible methods for compensation, or—better—executing the one that the user would prefer. It should also recognise that the user may have real-world adjustments which it cannot detect (e.g. the user putting on reading glasses).

Unfortunately, as will be discussed later, there is little data on the severity of impairments or how people cope with them. Therefore any system devised would have to, in theory, be prepared to cope with the whole range of observed human capabilities and large fluctuations within these. Data from projects in progress such as Sus-IT may be used in future to provide suitable default properties for
people in various situations (i.e. with various medical conditions or of a particular age).

2.8.1 Adaptive Accessibility Features in Computer Systems

Given that access needs vary both across and, temporally, within users, devices and environments, it would be reasonable to expect that contemporary computer systems provide some level of support for this. OSs expose functionality to applications by providing a set of library functions that these applications can call when required—this is an Application Programming Interface (API). Table 2.2 lists the availability of adaptive accessibility features for users and APIs for applications in three contemporary OSs. As can be seen from the table, there is actually very little provision for adaptive accessibility features—though there is often some provision for more static accessibility features, as will be discussed in section 2.10.

2.8.2 The Notion of Disability

As highlighted in subsection 2.5.2, the concept of accessibility is strongly linked to recognised disabilities. As has been and will continue to be argued in this work, however, accessibility issues are far wider-reaching than this sector of society. However, it is useful to have an understanding of the ways disability has been defined in the past, so that we might better understand the current landscape of the AT industry and research.
Table 2.2: Comparison of adaptive accessibility features in common OSs (latest versions; including bundled applications)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mac OS X</th>
<th>GNOME</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sharing users’ settings across machines</td>
<td>●</td>
<td>●</td>
<td>●*</td>
</tr>
<tr>
<td>Apply users’ preferred settings on log-in</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Suggesting appropriate assistive features to user</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Checking appropriateness of enabled assistive features</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>API for informing applications of users’ needs</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>API for applying adaptations to applications’ content</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>API for detecting environmental attributes</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>API for controlling application connectivity</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

**Symbols:** ● = yes; ○ = no

* Sharing an AT on Windows would usually require that the low-level components of the AT are installed on every machine the user may use. This can cause incompatibilities with certain graphical applications, affecting other users.
The medical model of disability considers ways to cure the detrimental effects brought about by physical or mental impairments (relative to accepted norms) in the individual. The social model characterises disability more as a failing of the surrounding society to properly address the variable needs of people within a wider population. Many legal definitions of disability and requirements for the provision of education and services for disabled people now exist, often in the form of requirements of compliance with established standards in respective sectors (see subsection 2.10.2).

A theme that is becoming increasingly popular, particularly since the awareness-raising of the United Nations (UN) year (1981) and decade (1983–1993) of people with disabilities, is that disability (a) may affect everyone at some point in their lives and (b) may be caused by environmental and other external factors. Functional models classify the possible functionality of a human and measure the individual based on their capabilities. Standard classifications, such as the World Health Organization (WHO) International Classification of Functioning, disability and health (ICF) may be used to enable us to express the relative capabilities of one person against another, as well as the effects of the environment or disease on capabilities [124].

As well as functional classification, another relevant concept for this work that applies to more than just those people with recognised disabilities is that of dynamic diversity, discussed in more detail in subsection 2.11.4.

In terms of the spectrum of capabilities, those people with recognised disabilities may be at one end of the spectrum, but the accessibility barriers they face and—importantly—many of the solutions for them can be successfully applied to help people who are experiencing minor-to-moderate barriers.

### 2.9 Contrasting Accessibility and Usability

There are multiple levels on which the relationship between “usability” and “accessibility” may be discussed; this section covers some key areas. Figure 2.5 depicts four alternatives for each type of relationship.

#### 2.9.1 Causal Relationships

It is largely agreed in the literature that if an interface, program or content is accessible—in the sense that it complies with established accessibility guidelines—this does not necessarily mean that it is also usable [89]. Perhaps more readily understandable, artefacts considered usable by some are not necessarily also accessible to others—this could be caused by multiple factors, including one or more
Figure 2.5: Several possible relationships between “usability” (dark) and “accessibility” (light). From top-left, clockwise: overlap; subset; equal and disjoint.

of the following.

- perceptual problems, such as artefacts being rendered in an inappropriate modality for a given user

- due to the interaction of the user’s ATs with the content or interface in question

- because of external factors affecting usage of the device in question.

This seems to imply that there is not a strong causal relationship between usability and accessibility (in terms of guidelines), though there is some evidence to suggest that the more usable an artefact is, the more accessible it is likely to be (and vice-versa). This is perhaps an indication that authors or developers with a deeper understanding of one field are more likely to be aware of the other and take steps to embrace it out of best practice.

Further, it is largely the case that artefacts are considered usable—rather than accessible—if they are only considered to have one of the two characteristics. This implies one or more of the following.

- Awareness of accessibility is still low,

- that accessibility is seen as a low priority or is too hard to implement,

- or that accessibility is a more specific quality than usability and, thus, harder to achieve for significant portions of the public at once.

All of these possibilities appear to be the case, to some degree. The third potentially indicates that accessibility “coverage” of the population may be improved by adopting more adaptive techniques for delivering interfaces and content.
A study by Sloan et. al. [99] showed that usability problems are likely to indicate accessibility problems—and cites other work that discusses the strong relationships between designing systems for disabled people in normal circumstances and designing systems for non-disabled people in more extreme circumstances [115, 80]. On this subject, it is interesting to note that the theme for W4A 2006 was “Building the Mobile Web: Rediscovering Accessibility?” [54], which highlights the acceptance of these links between accessibility and usability for users with and without recognised disabilities.

Finally: the “POUR principles” introduced in section 2.5 represent the basic requirements for users to be able to access systems in order to make any meaningful use of them. These principles were noted to appear more fundamental than the dimensions of usability introduced in section 2.1. Clearly the POUR principles are a pre-requisite for anyone to be able to use a system, not only those with recognised disabilities, even though the contemporary focus of accessibility is on those with recognised disabilities. The result of this is that accessibility in the sense of this thesis’ proposed definition—providing access for as many people as possible—can be considered as a pre-requisite for usability.

2.9.2 Effects on the Population

Until the previous section, both the contemporary definition of accessibility, which focusses on access for people with recognised disabilities and ATs, and the definition proposed for this work, which considers access for all, have been compatible. When discussing the effects accessibility barriers and usability problems can have on the population, however, they become incompatible, due to the different definitions of “the population” (one being a subset of the other).

The traditional view of accessibility, already introduced, is that it affects a subset of the population—i.e. those with recognised disabilities. As above, accessibility barriers are forecast to become more relevant to a wider range of people, partly due to the move to ubiquitous computing and, thus, the prevalence of “extreme” usage situations. From the perspective of a population of users, this means that accessibility barriers are likely to affect a larger proportion of these users, though by differing amounts and in differing ways—i.e. some users, by virtue of impairment or environment, will be more susceptible to some types of problems, at particular times, than others.

The view that accessibility concerns affect more people than was previously imagined is backed up by Petrie et. al. [87] as well as many others [38]. Petrie et. al. define accessibility and usability problems for users as distinct but overlapping sets and classify problems as: pure accessibility (only affecting those with disab-
ilities); pure usability (affecting only non-disabled users) and universal usability (affecting everyone). They take the contemporary definition of accessibility and correctly (under that definition) assert that there are accessibility barriers faced by people with recognised disabilities that others will never face (such as those introduced by colour perception deficit). They discuss that (under the contemporary definition) Thatcher et. al.’s assertion that accessibility barriers affect a subset of the population that are affected by usability problems [108] misses the recognised disabilities-specific barriers. However, the thrust of this chapter’s argument is that although there are clearly chronic difficulties faced by those with specific recognised disabilities, a more future-compatible definition of accessibility would cover the whole population (as anyone may—and many people do—experience barriers brought about by usage context and ageing [91, 50]).

In this case, the remit of accessibility and usability difficulties is equal—everyone, albeit with some people (either those with recognised disabilities, or in particularly extreme circumstances) will be affected more than others. It seems to be universally accepted that users’ usability experiences will vary and that usability testing should be carried out with a range of users. It is asserted that accessibility experiences, for the same reasons, should not be treated differently.

As a counter-example to the colour deficit argument, that the barrier is exclusively a disability matter: even “normal” users will be unable or unwilling to perceive some colour combinations, so although the issue may be considerably more about user preference (usability) than access to the system (accessibility), such matters should still be taken into consideration—and should a system offer adaptations or customisation opportunities in order to mitigate them, this can benefit non-disabled users too.

Finally: the temporal incidence of accessibility barriers is largely fairly static (or at least predictable) for people with recognised disabilities. However, for people who are largely experiencing problems caused by devices and environments, the barriers are less severe (as above) but also more likely to be intermittent. Still, however, their coverage is the population as a whole.

2.9.3 Usability Problems vs. Accessibility Barriers

When does a usability defect become an accessibility barrier? When is an accessibility barrier overcome and subsequently may be regarded as a usability matter? It has to be accepted that, given the variance in users’ and devices’ capabilities, a usability defect/accessibility barrier is a point on a continuum, much as the spectrum of users’ capabilities (Figure 2.3). That is to say: depending on the person, device and scenario, a usability matter may be seen as anything from a trivial an-
noyance, through being a hindrance to efficient workflow, to being a grave barrier to accessing a given system.

Further, should a perceived accessibility barrier (e.g. small text size) be compensated for (e.g. by enlarging the text), it may be impossible to accurately determine how much correction may be required to simply allow the user to perceive/operate the system and, thus, at which point the barrier to access became a usability variable requiring perhaps some further optimisation. Thus, accessibility barriers cannot be seen as binary.

Referring to the list of potential barriers from subsection 2.5.3, the difficulty in categorising each as an accessibility or usability problem, even for a given person at a given time, becomes apparent, as below.

**Reading text on a computer screen.** It is unclear what level of increase in size is necessary for access or improves usability (or, in the case of substantial size increases, may impair usability).

**A set of stairs.** A wheelchair user definitely faces an accessibility barrier, but at what point does someone who finds walking hard have to give up and find an alternative route?

**Reading information expressed in a given language.** The language must be known by the reader. If it is, then usability may still be impeded if the material is expressed using an outdated phrases or in overly-complex manner.

It is argued, however, that this classification difficulty is actually an artificial problem for the purposes of this work, which is focused on matching people with minor-to-moderate impairments to the appropriate forms of AT. In this setting, the user is often at least able to communicate to the system that there is some sort of problem—and, through either active or passive means of communication, is able to direct the solution. Essentially, the user should be treated as the expert part of the overall expert system.

Appendix A contains an example discussion of the usability and accessibility issues surrounding a contemporary mainstream application’s interface, using the adopted definitions of usability and accessibility. This discussion illustrates how, as the transition to more pervasive computing continues, the contemporary definitions (particularly that of accessibility) are becoming too restrictive.

### 2.9.4 Definitions Used by this Thesis

Carey [23] raises a fundamentally important point: that, surely, the only genuinely useful measure of a system’s ability to cater for its users is if it enables them to use
it for its intended purpose, regardless of if the users have recognised disabilities, regardless of if accessibility is defined as a continuum affecting the entire population or otherwise—arguments over the definitions of accessibility vs. usability are not terribly useful in the real world. The purpose, in some respects, of the previous section was to illustrate that although there are clearly areas of research focused on different areas, such as objective and subjective measures, rendering techniques and workflow optimisation, the ultimate effect of these on people attempting to access and use artefacts is much simpler, in line with Carey’s assertion.

It is argued that, for these reasons, much simpler, more portable and blunt, though indicative, metrics are required for determining “accessibility” and “usability” in terms of their effects on people at the highest level. Such metrics must ideally be population-based, because laws devised to ensure appropriate accessibility require metrics that are always relevant, as the laws themselves are effectively immutable. A metric with these properties is proposed in chapter 4.

2.10 Present Assistive Technology

In contrast to physical artefacts and devices, electronic systems may possess a great deal of flexibility; whereas the interface and internal mechanism of a totally mechanical device may be fixed (or at least prohibitively difficult for the user to modify), the interface, content and internal workings of an electronic system can be designed largely without such limitations.

A very simple, yet popular, instance of where this flexibility has been used to the advantage of users is in many OSs’ provision for a variable global font size and variable parameters for keyboard and mouse sensitivity, as well as double-click timings. These are all fundamental parameters that need to be considered by the OS regardless of whether they were exposed as settings to the user—but adding that exposure allows the system to be tailored to some extent to individual users. This may be seen as an example of “design for each”, albeit with the requirements that the user must be aware of their needs and change the settings appropriately, as touched upon in subsection 2.8.1.

Unfortunately there are many examples of where flexibility has not been employed to advantage users. Electronic document reading is a good example: though many people prefer not to read documents from a screen, those that do are often constrained by notions such as “the page” and fixed layout, rather than the semantics of the information they are dealing with. This may be a minor annoyance to people without print disabilities, but for those with such difficulties, or small-screened devices, such tasks can be very difficult (as discussed in relation to the assumptions made about the design of the Microsoft Office UI above).
Further, as with physical devices, computer interfaces are generally designed with one demographic in mind—unfortunately often the “average user”, who does not exist.\footnote{The (arithmetic) mean user does not exist; a “typical” user of course may exist, though this chapter has argued that DUET barriers are still almost inevitable, particularly as computing becomes ubiquitous.}

The result of this is that barriers to accessibility may be created, because the user is not able to change the way interfaces and content are presented to them—much the same problem that they may face with physical-world situations and devices. In response, several techniques, involving hardware, software and standards have been created to mitigate accessibility barriers. This section discusses the nature of typical contemporary AT; the following section critiques it.

### 2.10.1 Availability

On the desktop, most major platforms—the GNOME GUI environment;\footnote{No other GUI environment for non-Mac UNIX-like operating systems currently supports an in-built accessibility standard, though the K Desktop Environment (KDE) project is adopting the same standard as GNOME (AT-SPI). Source: \url{http://doc.trolltech.com/4.0/qt4-accessibility.html} (the Qt GUI toolkit forms the basis of KDE).} Mac OS X and Windows—now provide at least one GUI framework for developing applications that automatically exposes information required by assistive technologies (ATs). The standards employed are different for each GUI toolkit, though there have been some efforts to bridge these gaps, such as the effort to bridge Microsoft’s new “UI Automation” standard to that used by GNOME.\footnote{\url{http://www.mono-project.com/Accessibility:_Architecture}} Table 2.3 shows a brief overview of adaptations supported out-of-the-box\footnote{Microsoft describes Windows’ in-built “Narrator” as “intended to provide a minimum level of functionality for users with special needs” and that “most users with disabilities will need utility programs with more advanced functionality for daily use” [76]. The “Magnifier” tool, which is incapable of full-screen magnification, is described similarly [75].} and available as third-party add-ons in common operating systems.

Similar standards exist—and are used—for allowing ATs to access the Document Object Model (DOM) of web content in three popular browsers: Firefox (Mozilla), Safari (Apple) and Internet Explorer (Microsoft). This is particularly important as the Internet becomes a vital channel for social and commercial interaction.

In the smartphone and PDA arena, both screenreaders and magnifiers exist for devices running Symbian 60 and Microsoft Windows Mobile (at extra cost).
Table 2.3: Comparison of accessibility features in common operating systems (latest versions; including bundled applications)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Mac OS X</th>
<th>GNOME</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-screen magnification</td>
<td>I</td>
<td>I</td>
<td>C</td>
</tr>
<tr>
<td>Colour deficit support</td>
<td>P</td>
<td>P</td>
<td>C</td>
</tr>
<tr>
<td>Resolution and text size</td>
<td>P</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>Screenreader</td>
<td>I</td>
<td>I</td>
<td>F/C</td>
</tr>
<tr>
<td>Read specific text</td>
<td>I</td>
<td>F</td>
<td>C</td>
</tr>
<tr>
<td>On-screen keyboard</td>
<td>C</td>
<td>I</td>
<td>C</td>
</tr>
<tr>
<td>Voice recognition</td>
<td>P</td>
<td>C</td>
<td>I</td>
</tr>
</tbody>
</table>

Fields: I = Integrated; C = Commercial add-on; F = Free add-on; P = Partial support.

2.10.2 Standards

A number of standards have been developed to enable the development of ATs themselves, and then to optimise the accessibility of the applications and information to which ATs will facilitate access.

Most popular GUI toolkits expose, as discussed, accessibility information to ATs. There are many standards for this; some open and some proprietary. Recently Microsoft has proposed a new standard, “UI Automation”, to replace the previous Microsoft Active Accessibility (MSAA), though currently no commercial AT for Windows supports it.

Direct access to web content for ATs is provided in most popular browsers by bridges to (standard protocols for ATs to interrogate) the DOM.

Web content may be created in line with recommendations issued by the W3C [122]. The existence of accessibility standards provided by such an influential and respected group has helped increase awareness of web accessibility issues [30]. Accessibility has been incorporated into the activities of standards activists, due to it being part of the established best practices for ensuring the compatibility and usability of information published on the web.\footnote{See \url{http://www.webstandards.org/}.}

However, it should be noted that, because the overwhelmingly vast majority of web sites are \textit{not} standards-compliant [29], web browsing can still be difficult and may even require specialist browsers to be used for the time being.

Web applications now also benefit from industry standards [118].
Game accessibility guidelines have been created by groups such as IGDA, DIGRA and independent researchers as part of an awareness-raising effort in the computer games industry—sometimes presented in engaging ways to developers [49].

Laws have been introduced with the aim of ensuring that disabled people are not discriminated against and that technology used by government is accessible to as many citizens as possible—for example: the Section 508 standard in the US.

2.10.3 Unanticipated Effects

Sometimes when an “accessibility” technology is developed, it can be of use in unexpected ways. A prime example of this is the use of accessibility APIs by graphical program testing suites, such as the Linux Desktop Testing Project or Strongwind. These systems allow developers to test their GUIs by simulating interactions from a real user and ascertaining that the GUI is in the expected state after such interactions.

Integrating an “accessibility” technique into a mainstream system can benefit more people than it first may seem. After the success of the first game in the “Half-Life” series, Valve was asked to add closed-captioning to allow gamers with auditory disabilities to play the game. According to 1997 census data used by IGDA, 3.8% of the population of the US that were over 15 years of age experience moderate or severe auditory problems. Valve implemented closed captions for Half-Life 2 and were also able to use the captioning infrastructure to market the game in countries for which no vocal dialogue (used extensively to convey plot events) had been recorded. It took only two weeks of one developer’s time to create the infrastructure for closed captioning, plus some further unspecified time to tune it. Valve has included this facility in its engine, which is available to licence.

A counter-example, in which the omission of an accessibility feature has caused problems for a wide range of people, can be found in the iPhone from Apple. Though this device represents a step-change in terms of usability—taking the idea of direct manipulation much further than other contemporary interfaces—there is a striking omission with respect to accessibility: the lack of a global minimum font size option. Such an option does exist on most desktop operating systems.

---

16http://www.section508.gov/index.cfm?FuseAction=content&ID=12
17http://ldtp.freedesktop.org/wiki/About/
18http://medsphere.org/community/project/strongwind/
19http://developer.valvesoftware.com/wiki/Closed_Captions
(other than the Mac OS) and many mobile telephones. The lack of this feature has caused application developers to have to implement it themselves, even in applications that use the iPhone’s native GUI toolkit (Cocoa Touch).

The reason this feature is a striking omission is that it affects many people who do not regard themselves as being disabled, or elderly. Feature requests for iPhone software, questions on how to change the font sizes and articles on how to compensate for these problems are plentiful. Since the initial product launch in 2007, certain traditional ATs have been added—namely a screenreader and screen magnifier with colour-inversion capabilities. These additions are laudable, however they do not adequately solve the legibility problem for the general public. This is because the ATs are marketed specifically towards users with disabilities; most users are likely to be unaware of them and they represent significantly more complex solutions than providing a minimum font size setting in the first instance. Unfortunately, this has not been remedied in later devices, such as the iPad.

2.11 Problems with Present Assistive Technology

Though a great deal has been achieved in furthering opportunities for people with disabilities in the electronic world, a great deal of work needs to be done to achieve more widespread recognition and use of ATs. Many of the problems can be partially blamed upon the retro-fitting of accessibility concerns; it is argued that if accessibility was considered throughout the design process, artefacts would be more suitable for this work’s target audience—those experiencing minor-to-moderate barriers—but that this would also improve the situation for people with recognised disabilities who are currently regarded as the AT industry’s main market.

2.11.1 Undiscoverability and Immobility

It is also important to consider how discoverable and available a given AT is to the user it is intended to support. Currently, users must go through several steps: (a) awareness of their need for an AT; (b) awareness that such an AT might actually exist; (c) procurement; (d) installation and configuration and (d) usage (including any adjustments in configuration). Some of these problems may be addressed in

---


22Accessibility features remain the same ([http://www.apple.com/uk/ipad/specs/](http://www.apple.com/uk/ipad/specs/)) and no global minimum font size setting has been added.
domain-specific solutions [18]. These matters are discussed further in the following chapter.

Perhaps the key issue of AT discoverability is that of mistaken user identity (by both the user and developer). Even when in-built accessibility features are provided by an operating system, they might be grouped under the banner of “disability access”, often indicated by the internationally-recognised wheelchair icon. Though the inclusion of such tools is laudable, their intended audience is often misunderstood by the general public as not for them; they do not have a recognised disability. In fact, these tools are often aimed precisely at those who do not need a full-featured (and expensive) access technology and thus only need incidental help some of the time.

Even when ATs are installed, configured and in use, they can pose usability problems of their own [121]; this indicates that AT developers might need to focus more on the usability of their products, as well as re-evaluating the true size of the AT market, as will continue to be discussed throughout this chapter.

2.11.2 Indirection

Retro-fitting certainly does not help matters such as consistency and discoverability. It also introduces some fundamental constraints on both the potential of ATs to assist users and burdens (particularly cognitive) on users trying to make use of them to access content and applications. Perhaps the most serious of these problems is that of the indirection introduced when layering ATs on top of other systems. Users of screenreaders, for example, are required to understand the model of the underlying GUI-based system, which is a daunting task in itself, but is made harder by the usability problems inherent in ATs themselves [15].

Contradicting views of this redirection were introduced in subsection 2.4.3 and then again in subsection 2.4.5. The contradiction was that, on the one hand, forcing users who are not capable of perceiving a GUI to be aware of it and its terminology was considered detrimental, yet Apple’s VoiceOver approach to allowing a blind person and a sighted person to collaborate on the same device was praised.

The reason for the praise was that the VoiceOver approach helps to provide a common frame of reference for both users. A common frame of reference is required in order for people to be able to interact. If two users are interacting via remote devices, it may be possible for them to use radically different interfaces, as long as both interfaces supported the underlying task. However, if two people are using the same local device, they must both be able to perceive and understand the interface, which requires them to share a similar frame of reference (i.e. GUI terminology
Simply due to the fact that most contemporary “computers” use the GUI paradigm, it is the lowest common denominator and this mandates that even users of TTS and Braille, via ATs are required to understand it. Given this, however, VoiceOver provides a way for vastly-improved collaboration with other users.

2.11.3 Rigidity of Underlying Systems

Returning to the problem of reading electronic documents from section 2.10: the inflexible nature of some document-reading systems can limit the potential of the electronic system, which ends up being nothing more than a digital clone of the analogue artefact. For users who require more flexibility of presentation, although ATs may be introduced with the intention of addressing those problems, they are often unable to extract the information required in order to re-present the content in an accessible way \[73\]. The underlying problem in such situations is that either the AT is unable to access the source data (due to it being in a separate process, for example) or, more seriously, that the source data no longer exists. In the case of screenreaders attempting to make PDF files accessible, there is no source text in either image- or totally vector-based documents. This fundamental problem is very common amongst any retro-fitted technology and AT is no exception. Some further examples follow.

Use of non-standard widgets, as well as non-standard use of widgets, causes problems for ATs, as they must recognise each widget and be able to query the Operating Environment (OE) for properties of that widget. Some applications use custom widgets that do not implement the APIs that provide accessibility information, causing, for example, ATs to go to such lengths as using Optical Character Recognition (OCR) to identify data such as button labels. Inaccessible non-standard uses of widgets include the use of listboxes to provide the semantics of tabs in dialogue boxes. This is achieved by linking the listbox selection to code which hides/shows a group of widgets elsewhere in the dialogue box, which would appear to have a similar visual effect to switching between tabs (one set of controls replaces another). It is almost certainly an attempt to make a more attractive and usable alternative to the standard tab-style dialogue—but as listboxes were not designed to group other widgets, the semantic relationship is lost on a mechanical screenreader and the result for someone using AT to browse the UI can be confusing.

Rigidity of Legacy UIs is also an important factor. Even when data can be extracted from a legacy UI (e.g. via Assistive Technology Service Provider Interface (AT-SPI) or MSAA), there are limitations on how it can be (re-)presented to the
user. Screenreaders often use TTS and Braille output, whereas screen magnifiers effectively enlarge every pixel on the screen and may also provide contrast and colour controls. The problem is that it is generally not possible to change the appearance of individual widgets or groups of widgets, because the toolkit provided by the OE was almost certainly not designed with such mutability at run-time in mind. For example: adaptation plug-ins for dyslexic users would be unable to instruct the OE to change the label on a button. Though traditional users of screenreaders may not find this a problem, it is clear that the all-or-nothing approach to present ATs would not suit the minor-to-moderate and transient impairments experienced by the wider population.

It is now perhaps more clear that, when dealing with legacy applications, it is easier to make adaptations in modalities that were not specified as part of the original UI. Adaptations, such as wholesale enlargement of an application’s window, perhaps with some post-processing to lower the contrast, can be made, but they have to be made at a coarse resolution.

Fortunately, some applications are more extensible and provide methods for modifying various properties of their interfaces at run-time; examples include Microsoft Word and Mozilla Firefox—which provides a high-level, dynamic interface language, XUL. Such programs could be retrofitted by way of custom extensions, to connect them to an adaptivity system in the OE at a higher level, thus permitting more fine-grained UI adaptations (such as using TTS to read aloud content, whilst UI elements are simply magnified or left unaltered).

2.11.4 Low Recognition of Dynamic Diversity

The problems of rigidity do not rest solely with developers of mainstream applications; they are also present in some ATs themselves [15, 106], which appear to have been designed in some cases without adherence to established usability best practice.

Though the Universal Design approach must be used with physical artefacts, to give the greatest number of people the greatest chance of being able to interact with a device, this is not necessary in an electronic system. It would be incorrect to assume that a single user has a single disability and that a single AT would be sufficient to overcome the accessibility barriers they face. For the reasons discussed above, most people’s accessibility needs will almost certainly change over time and vary in number.

\footnote{As discussed, cognitive disabilities also place extra burden on the content provider, partially due to limitations such as those discussed here, but also due to the in-depth semantic knowledge that content providers, as opposed to outsiders, have.}
CHAPTER 2. LITERATURE REVIEW

It is important to recognise that capabilities fluctuate over multiple timescales [50] and that such fluctuations may produce unforeseen interference effects [71] (for example: difficulties caused by sensory impairments can increase cognitive load, reducing the resources available for people to reason about navigation and other matters). Further, the user's capabilities may be enhanced or diminished or enhanced by additional factors that may be difficult or impossible to measure directly, such as the availability of hearing or reading aids, or medication. Gregor et. al.'s term “dynamic diversity” refers to this interacting system of variables and fluctuations and can be applied to people from many backgrounds, over multiple timescales [22]. Of course this diversity also now applies to both the environments in which we use computers and the range of devices on offer.

As discussed above, there are many people who could benefit from improved computer accessibility—particularly adaptivity—because their capabilities vary over time. There also exist some standard models of human capabilities [35], including some data on how they are affected by disabilities [21]. However these two approaches have not yet been fully combined.

There is currently a dearth of detailed and long-term studies covering the needs of both disabled and non-disabled users who may experience minor to moderate difficulties (sometimes many simultaneously)—i.e. into the effects of dynamic diversity on AT needs as discussed above.

2.11.5 Inconsistency

Possibly due to the reasons above of expense and undiscoverability—either brought about or exacerbated by the fact that AT is often retro-fitted to existing systems, rather than designed-in, there is a high degree of inconsistency with respect to where AT is provided and what form it may take.

Consider a user wishing to access information on the web as an example; it is possible that there may be some AT installed on the user's machine and also that the content author may provide adaptive features within the web pages concerned (such as “speak this page”, text enlarging or colour-changing widgets). Figure 2.6 illustrates the four possible situations regarding AT availability.

Given that the user would be required to use a different method to employ any particular accessibility feature depending on whether it was present in their installed AT or, if within the page, how the content author created that page, the situation is clearly unnecessarily complicated.

Unfortunately the AT community has yet to offer suitable advice on this matter—proponents of page-based accessibility tools argue that it is a sign of content authors stepping up to the challenge of providing accessibility and is therefore
welcome. Additionally, given that a large number of people may not realise that AT exists, yet would benefit from it [100, 107], the low barrier to entry could be of great benefit. The relevant standards body, W3C, generally recommends that pages be designed for the lowest-common-denominator users and adapted accordingly by the client. Others claim that providing such features within pages effectively takes away the user’s opportunity to learn browser-, system- or AT-based techniques that would enable them to access any (standards-compliant) web site, or even other applications on their system such as word processors. The inconsistent provision of such features may lead users to errantly believe that only pages with such additions can be made accessible to them.

This raises the point of unequal support for ATs in various applications and particularly games; as an application (or game) uses technologies further removed from the standard APIs provided by the OS, there is a decreasing chance that an AT will be able to interrogate said application to provide improved accessibility to the user.

The difficulties surrounding web accessibility and inconsistency are addressed by the IBM Web Adaptation Technology (WAT) [91], in which common accessibility settings are exposed through a browser toolbar and changes to these affect OS accessibility settings. However as it is specific to web-browsing, this does not provide a system-wide approach to unifying AT configuration.

**Goal 6** (Consistency of Interaction with Assistive Technology). *Provide a unified mechanism for taking advantage of accessibility features, in much the same way that WIMP-style interfaces provide a unified mechanism for command discovery and actuation in GUIs.*

### 2.11.6 Lack of Accessibility in Authoring Tools

It is important to note that whilst a lot of work has gone into content accessibility, the general level of interest in authoring tool accessibility is low. This presents arguably even more serious barriers to people with disabilities.
2.11.7 General Lack of Industry Adoption

As discussed, accessibility, despite the legal and charitable aspects, often comes down to being a business decision on the part of application and content developers—the delivery of accessibility can never rest solely on the shoulders of AT vendors.\(^{24}\)

In very simple terms, achieving mainstream industry adoption of accessibility techniques requires methods that make it (a) easier and (b) cheaper for developers to do what they currently do whilst also providing accessibility features and ensuring compliance with the relevant standards.

By way of example: a research project into adaptivity may use a given system for storing and maintaining users’ preferences, which in turn uses one of a myriad different machine-learning algorithms. There will also be a user model, possibly quite specific to the adaptation being developed—and every coder has their favourite language. It is clear how it can sometimes be very hard to turn these proofs-of-concept into mainstream products—though specialist markets may be reached, there is little incentive for mainstream developers to spend the necessary implementation time on them.

2.12 Adaptivity as a Means to Mainstream Accessibility

One key problem is that of providing incentive for developers and content creators to make their work accessible. This is partly addressed by recognising that accessibility concerns apply to many more people than most initially believe—including many adults [38, 69] and older users [91, 51]. The argument for better support of older people is that they often experience capability fluctuations and decline due to the ageing process—and may also experience multiple disabilities at the same time [51]. Under this much wider and more realistic definition of who is affected by accessibility, it is in content providers’ interests to make their work accessible, particularly as a growing proportion of society will benefit [46].

Further, yet more people may experience “disabilities” brought about by limitations in the devices in use (e.g. small-screen PDAs, netbooks), or imposed by the environments in which they find themselves (e.g. particularly bright or noisy places). These effects, as with many of those brought about by the ageing process, may well be minor and temporary, but they do exist and could, potentially, be addressed using the same types of technology developed for those with more

\(^{24}\)It *cannot* because even with the inclusion of “scripts” and “maps” to improve application accessibility, third-party software cannot fundamentally change the behaviour or presentation of most common applications. It *should* not because accessibility is also a social concern that affects many people and, as such, should be an expectation of society as a whole.
severe impairments—e.g. TTS, key debouncing, content filtering, magnification. IBM’s WAT research showed that a surprisingly large number of users may opt for speech output when the opportunity is presented to them, despite this being considered an accessibility feature for people with disabilities [91]. The difference between this example and the rather static nature in which current commercial ATs are used is in the extent to which these techniques would need to be applied and the duration for which they would be required.

Due to the fact that people’s perceptions of an interface and the situations in which it is used are variable, both between users and over time for each user, it is clear that no single interface could satisfy the needs of all users—though attempting to design interfaces that have reasonable defaults and can satisfy most users “out-of-the-box” is a laudable goal (introduced earlier as “Universal Design” [109]). At a fundamental level, some of the content and other standards discussed in subsection 2.10.2 do provide reasonable baseline values for users’ capabilities [35]. Adopting these standards increases the likelihood that a given accessibility adaptation will be able to render any given content accessible for a user with differing capabilities because standard methods for adaptation may be used (such as the DOM [91]). It is of key importance, however, to note that adopting such baseline standards is just the beginning—doing so provides a common platform for adaptations to build upon but, for the reasons discussed throughout this chapter, this should never be considered the final step in ensuring accessibility.

As one of this thesis’ and Sus-IT’s goals is to reduce the effort on industry-based software developers, let it be assumed that their involvement should cease after designing their application and interface (ideally using a technique not dissimilar to their current one). However, there is clearly a lot of work necessary to provide the dynamic adaptive behaviour discussed above.

### 2.12.1 Variety and Configuration of Adaptations

On the one hand a range of adaptations is required: from those for motor-impaired users [43] to people with colour deficit [66]. It is also necessary to detect problems that the user may be having, so that these adaptations may be brought in only when needed—this could be when the user enters a noisy environment (simple to detect on most portable hardware) or it could be in relation to disability, such as motor control problems [62]. Finally, the configuration of the adaptations and the system controlling them is an important and complex matter. Trade-offs exist between the aggressiveness of the system in employing adaptations and the user’s sense of control, as well as issues surrounding the privacy of configuration information, which could be used to infer personal and possibly identifying details.
about the user [110].

2.12.2 Accepting, Exposing and Sensibly Applying Adaptations

Though it has been hinted at in some of the literature, there has been little practical work in the area of applying adaptations (a) system-wide—i.e. across a range of applications rather than in one particular application, allowing lessons learnt about the user’s needs in one application (such as contrast ratios and text size) to benefit others and (b) to enable the interfaces of applications to be expressed in a wide range of other modalities.

The concepts of personal universal controllers [82] and universal remote consoles [126], introduced in section section 2.4 attempt to tackle this; standards are defined for devices to express their functionality and software running on a PDA or mobile telephone presents an appropriate interface to the user to control a device. Currently only GUI and TTS interfaces are commonly provided. Though such projects are pioneering, the same modality restrictions are present in other multi-modal interface systems [92]. Systems that target multiple platforms that support GUIs can provide improved productivity to many users, but anyone who cannot access GUIs would not be able to benefit (as covered in subsection 2.11.2: using an assistive technology to access a GUI’s representation of the underlying interface is far from optimal).

IBM’s WAT [91] is able to make changes to system accessibility settings based on both detected user problems (such as tremors whilst typing) and at the request of the user. The way accessibility settings are presented to the user—and applied—is interesting, for two reasons. The many accessibility-related settings buried deep within the operating system are brought into the WAT in-browser interface so that users can easily find and change them. Further, the settings are applied not just to web content but the application as well—i.e. a text size increase will affect the browser’s menus too. This is almost certainly what the user would wish for, but the literature yields very few examples of this sort of approach—and Richards et. al. found that older people who do not identify themselves as disabled found it useful [91, sec. 3].

In a system designed to cope with fluctuating user capabilities, device and environmental constraints, there is a possibility that two conflicting adaptations may be required at the same time. Though sometimes there are solutions to this (such as swapping full-screen magnification for GUI widget enlargement in systems

---

25application-specific usage/preferences data could be held by the operating system in much the same way that settings for individual window display are now.
designed for users who are both vision- and motor-impaired [40, fig. 5.4, p. 113]), there has been little work on what the general case—and solution—may be.

### 2.13 Encouraging Industry Adoption

The work discussed above sought to provide examples of how accessibility can benefit a greater number of people than developers may initially believe. Though this may improve the business case for implementing accessibility, it still may not be enough to encourage developers to do so.

A major barrier is that any industry with established working practices will be averse to change—and the tools currently being used to build interfaces do not support adaptivity well. Though some tools²⁶ support the provision of accessibility information to external ATs, it is not compulsory to provide such information.²⁷ Further, it has been established that adaptivity is a key requirement and no mainstream interface toolkits support this. It is clear, therefore, that either: (a) research into AT (and HCI) proposes new development techniques that are perhaps radically different from those of today, but incorporate accessibility or, most likely in the short-to-medium term, (b) ways are developed to integrate accessibility—and, more importantly, adaptivity—into contemporary development environments.

If accessibility were an automatic side-effect of using existing development tools, it is unlikely anyone would avoid it. Though this goal remains distant, recent work has sought to begin bridging that gap.

Several standards exist for abstract mark-up of user interfaces—many based on XML—though only some may be suitable for use in adaptive systems [112]. There are a number of key requirements for such interfaces [125, 81, 112], namely the ability to be targeted at a wide range of devices (from desktop computers to washing machines) and delivery contexts (users with certain capabilities in a given environment); providing different means of accessing commands (e.g. direct access for expert users vs. methods for discovering available commands for novices); the ability to be rendered in a number of modalities (taking away some control over the rendering process from the developer); to be personalisable and mutable at run-time (as circumstances on the part of the user or environment change).

As was touched upon in section 2.4, it should be noted that although there do

²⁶ Microsoft’s “Visual Studio”, Apple’s XCode “Interface Builder” and GNOME’s “Glade” GUI builder
²⁷ e.g. though labels can be used by screenreaders to read out the intent of a GUI control, it is not possible for them to automatically ascertain the dependencies between controls such as dual password-entry boxes.
exist some popular mainstream techniques for describing interfaces with XML,\footnote{Most notable would be XUL from Mozilla and XAML from Microsoft, though some development tools, such as GNOME’s Glade, have been storing GUI definitions as XML files for some time.} these are currently focused only on designing GUIs, which is evident from the vocabulary of available widgets.\footnote{e.g. “button”, “toolbar”, “menu”.}

Gajos’ decision-theoretic interfaces provide a framework for adaptive abstract interfaces has been used to transform the abstract UI into a concrete version, whilst taking usage data and the user’s capabilities into account [40, chapter 3]. This work also proposes that plug-ins for contemporary development tools should be produced that allow the computer to infer the abstract interface specification from the actions of a developer designing the GUI. Currently, however, adopting truly abstract (possibly XML-based) interface mark-up would require considerable re-training of developers, so is unlikely to happen until the technology matures and offers further perceived business benefits.

A project that seeks to create a Global Public Inclusive Infrastructure (GPII) to aid mass-adoption of ATs exists and the reasoning processes developed here provide drop-in candidates for the proposed GPII’s “match maker” and “preferences storage” components,\footnote{http://gpii.net/components} thus increasing their potential usefulness in the eyes of platform and AT vendors.

As noted by Vanderheiden\footnote{[114]}, an open market for adaptations could be created (an effective “Accessibility App Store”—though it is asserted that this should be tightly integrated into the host platform, rather than segregated). AT developers could increase their potential market massively by moving to micro-ATs based on user capabilities. Platform vendors would intrinsically gain improved access for a more diverse user population, which would improve as more adaptations are developed.

\subsection{Configuration and Security Matters}

Privacy and security are extremely important in any adaptive system\footnote{[110].} The design of the techniques developed in this thesis takes these matters into consideration. However, the majority of the work on ensuring users’ data is handled sensitively and securely is being carried out by the wider Sus-IT project. For information, design decisions relating to security (amongst the wider issues of process architecture) are discussed in Appendix B.
2.13.2 Criticisms of Adaptivity Revisited

Some substantial criticisms of adaptive interfaces were introduced in section 2.3, yet an adaptive solution is being proposed here. In fact even Dieterich et. al.’s review cites work which explains that user needs adaptations should be under the control of the user [28]. The concerns raised will be addressed directly in the following chapter—but it is important to bear in mind that an adaptive accessibility system is different from an adaptive interface in the following ways.

- Classic adaptive systems were developed to improve usability matters; i.e. make a number of changes in order to optimise the experience. Accessibility barriers are problems that interfere with a person’s usage of an artefact. It is hypothesised that users would be more amenable to such adaptations, particularly if the adaptations improved access to a given system. This relates strongly to users’ appreciation of “caring technologies”.

- As the focus is on users with low awareness of ATs but with generally minor-to-moderate impairments, it should be possible to communicate with the user in order to ascertain when adaptations may be necessary and if executed adaptations have been accepted by the user. Clearly communication specifically regarding ATs should be minimised; the thrust of this thesis is on developing techniques to help ensure this.

- Though some adaptations will be more apparent or jarring than others (e.g. reading the content of a document via TTS vs. slightly adjusting the font size of an application’s UI), it is hypothesised that the most common adaptations will be (a) smaller ones applied over large timescales that (b) the user requested (either directly or via the system, based on a learnt user model). If, for example, a user always increases the default zoom level when opening a wordprocessor document (possibly due to a particularly high-resolution monitor on a particular device), then the system may learn this and apply that adaptation in future on that device.

- Many adaptations may be applied “behind the scenes”—i.e. before content is actually displayed to the user—so that, from the user’s point-of-view, no noticeable adaptation has taken place. Examples may include minimum font sizes on web pages (which are currently handled this way by most browsers) and, looking to the future, automatic rendering of pictures to compensate for any colour perception deficit on the user’s part.

---

31 This contrasts with people who experience chronic and severe impairments, who may be less able to communicate their needs—but whose needs would hopefully be better understood by others, due to their severity.

32 Again, the nature and challenges of such an approach are discussed in depth later.
Some proof-of-concept testing for treating the user as the expert on their own accessibility matters has been carried out and is documented in chapter 9. It is important to bear in mind that Sus-IT has provided the social motivation for this technical approach (via prior and on-going participatory research) and is carrying out various forms of pilot and longitudinal testing with real people to test these hypotheses over time and a larger sample of users.

Though it has been argued that adaptivity has great potential to improve accessibility— and that changes should be accepted when they are considered necessary by both the user and system—it must be borne in mind the unnecessary (or overly-aggressive) changes should still be avoided when possible. Particularly in the case of an adaptive system based on an abstract interface, which by its nature would be highly mutable, it is important to ensure that changes are only made: (a) when necessary and (b) with some awareness of predicted upcoming changes. This leads on to the introduction of a goal that provides a counter-balance to Goal 4, Temporal flexibility, as follows.

Goal 7 (Temporal balance). When making adaptations, aim to make the fewest number possible over a given time period, to ensure the stability of the system.\(^{33}\)

### 2.14 Summary

A number of related areas of research and commercial development have been discussed in order to highlight some of the positive impact that awareness of accessibility matters has had, as well as some key misconceptions, failings and opportunities for furthering accessibility in the emerging world of ubiquitous interfaces. Several goals for ensuring accessibility have been highlighted; in this section the remaining general cases where these goals have not been fully realised are discussed and the research questions for this work are defined.

#### 2.14.1 Commercial and Research Assistive Technology

Research has shown that there is no such thing as “the average user” [69] and that many people could benefit from some form of assistive technology [38, 53]. Academics have independently developed solutions to specific accessibility problems—those faced by a particular disability group or people in a certain situation, such as having colour deficit [66]. Due to being scientifically-validated projects, they must often be implemented using specific, controllable, technologies (e.g. using specific languages and tunable libraries or run-time systems) and are therefore difficult for industry to adapt into products.

\(^{33}\)Stability is discussed in section 7.4.
Commercially-available ATs have gone a long way towards removing numerous large-scale accessibility barriers for those with specific and often severe disabilities such as blindness and severe motor control difficulties. However, due to the focus on those with severe impairments, the market is small and there is low awareness of accessibility barriers and solutions within the general population. Further, it should be noted that more content-oriented disabilities, such as dyslexia and hearing impairments have only been addressed in specific cases by commercial AT because the solutions to these problems often require content authors’ direct attention.

As a result of the nature of commercial AT, people experiencing intermittent or gradual accessibility difficulties experience the problems described in subsection 2.11.1 and there is little support for such problems in mainstream OSs—i.e. no system-wide APIs for content and UI adaptation, or for reasoning with DUETS constraints.

2.14.2 Adaptivity for Accessibility

Adaptive interfaces for the purposes of improving accessibility have become the subject of extensive research [110, 66, 40]. However, several key challenges remain, particularly considering the trend to ubiquitous interfaces and interaction, as follows.

Most current adaptive interfaces for accessibility consider either only adaptations made to GUIs or perhaps providing GUI and TTS versions of a given interface, as in GADEA [92], HOMER [93] and the PUC [82], where interfaces are generated from an abstract description of an appliance’s functionality. Few if any projects consider the full range of modalities available for interaction. Further, no known research exists on the possibilities of composing desktop interfaces out of elements of multiple modalities.

Some studies take into account that accessibility improvements may help older people, as opposed to specific disability groups—and that, as with some notable universally-designed products discussed earlier, have gone on to be used beyond the expected target audience [53]. Unfortunately, projects such as WAT, Gajos’ SUPPLE system, GADEA and the pioneering adaptive systems detailed earlier are mostly focussed at application-level (such as web browsing), as opposed to applying changes system-wide.

The work of Gajos [40], provides a powerful example of how adaptive interfaces can be effective and well received and is a system that is largely targetted at accessibility improvements (via SUPPLE) and general usage patterns (via ARNAULD). The system developed can adapt over time to changes in user capability and pref-
references (Goal 4, Temporal flexibility) and allows the user to concentrate on using the application rather than the AT. However, there are some shortcomings of the approach taken by this and other contemporary work, as follows.

**Environmental factors** and their effects on users’ or devices’ capabilities are not considered—i.e. Goal 2, Accessibility barrier causes is not fully met.

**Run-time mutability.** Though the user interfaces may be generated very efficiently and the system has been designed to enable devices with low computational abilities, such as mobile telephones and PDAs to benefit from adapted interfaces, these interfaces are not mutable once instantiated. Though the interfaces adapt to historical usage and capability calibration data, they cannot react to sudden or predicted changes in user capability on-the-fly—i.e. Goal 4, Temporal flexibility needs to be more fully met.

**True multimodality.** In SUPPLE, adaptations have been employed to mitigate accessibility barriers for those with motor control impairments. Some basic adaptations have also been made for those with vision impairments (including reorganising of the interface to account for lost screen-space caused by these adaptations). However other modalities and disabilities are not considered. Similarly, other projects such as GADEA and the PUC have created GUI and TTS interfaces for people with vision impairments. HOMER allows for visual (GUI) and non-visual interfaces, through the use of a “rooms” navigation metaphor, 3D audio and braille. Few, if any, projects consider more than two target groups of people (those without a recognised disability and those with one in particular).

**Multiple optimality.** Many contemporary projects (such as SUPPLE, GADEA and PUCs, as discussed) take the view that there is one “correct” interface for a given user and device. This is sometimes implicit due the work beginning with an existing GUI and making adaptations to it in either the graphical or other modalities, as with GADEA. Other work that begins with an abstract interface definition frequently concentrates on transforming this into a single concrete GUI, as is the case for decision-theoretic generation used by SUPPLE.

However, when an interface is rendered across multiple modalities, devices and timeframes, one must be open to the possibility that there could be alternative renderings that could satisfy the user’s access requirements. In reality the user may prefer one style of rendering—particular mix of modalities, devices—to another, but in terms of mitigating accessibility barriers, multiple interfaces may present equally-valid solutions.
Failure. In some cases it may not be possible to make an appropriate adaptation to the interface. Though considerable research effort is put into developing new accessibility techniques, these may not always work. When such a situation arises, there should be an accessible way of alerting users to the fact that some features of the application in use are not available to them (perhaps unless they change their device or location). Ideally there should be a way for users in these situations to solicit help from other users. See Goal 5, Compensating for accessibility barriers.

This gives rise to the following goals that have not yet been met in concert.

**Goal 8 (Multiple optimalility).** Support the notion of multiple optimal interfaces existing for a given situation, possibly in different devices and modalities, and combinations thereof.

### 2.14.3 General Adaptive Interfaces

As discussed in subsection 2.2.3, historically, adaptive interfaces have focussed on areas such as portability and abstraction for the convenience of the developer, protecting them from platform idiosyncrasies and ensuring the user is presented with semantically-valid options at any given time and facilitating the provision of context-sensitive help based on the current task in progress and application state [105, 59]. Since its beginnings with context help systems in the early 1990s, the field of intelligent user interfaces has evolved out of the early attempts to produce more “helpful” interfaces for users. Typically, interfaces classed as “intelligent”, including recommender systems, programming by example and computer-generated opponents in games require deep knowledge of the problem domain to be successful. This makes some of the techniques used unsuitable for wider employment in more generic interface systems (such as the interface toolkits provided by a general-purpose OS). Further, recent commercial attempts at using generic adaptive interface techniques to improve usability have not met with great success—see subsection 2.3.1.

It is surmised that an increased focus on accessibility issues in adaptive systems of the future would be welcomed by users—the phenomenon of “caring technology”—and would help more people who could benefit from improved accessibility do so (solving the issues of undiscoverability discussed in subsection 2.11.1). Techniques are developed in subsequent chapters to glean and successfully encode as much knowledge as possible from the users themselves in order to make this practicable.

---

34 which are often capable of adjusting their skill level and tactics according to the player’s abilities
2.14.4 Distributed Interfaces

As the age of ubiquitous computing dawns, the strong possibility arises that interfaces will be (a) targeted at multiple different types of device [1] and (b) spread across devices [94].

Accessibility issues seem to be an under-researched area in this field, though due to the somewhat obvious limitations of some popular devices even for users without recognized disabilities, there have been high-profile discussions as to how to address these problems—parallels have particularly been drawn with respect to matters of web access on small-screen devices [54].

The research area of Ambient Assisted Living (AAL), which takes influences from such diverse fields as ubiquitous computing, AI, healthcare security, ethics and knowledge management, has now matured to such a level that large-scale international research projects are being mounted [31]. Many of these projects have been motivated by the phenomenon of population ageing [46].

2.14.5 Collaborative Interfaces

Research and development in the field of CSCW has generally not considered accessibility, hence there is a recognized need for collaborative interfaces to be made more accessible [93]. There is already a tendency towards “social accessibility” solutions [95] and, as the user base of ICTs grows, differences in cultural backgrounds as well as users’ mental models of technology must be taken into consideration [61].

Web-based collaboration tools—particularly those with a social emphasis—are becoming increasingly common and pose the additional problems of access to the pages generated by such web applications as well as to the collaboration itself. For this reason, W3C’s ARIA standard [118] for providing access to changing information on web pages has been developed. Unfortunately, if past take-up of web standards and particularly web accessibility standards is to be considered, it is unlikely this will be implemented widely by web developers.

Two primary types of collaboration have been highlighted by literature raising the need for accessible CSCW tools [93]: (a) local collaboration, in which more than one user interacts with the application together on a single, shared, device and (b) remote collaboration, where users collaborate from remote locations, each using their own device. These styles of interaction are depicted in Figure 2.7.

However, considering the possibility that the collaborative application has a distributed interface—as is increasingly likely—then there are two additional general possibilities for the collaboration: (a) that both users have a separate set of devices being used to access the collaboration and (b) that both users have their
Figure 2.7: Local and remote collaboration scenarios. Top: users sharing a computer; bottom: users with individual computers.

own set of devices and some of these are shared between users. These possibilities are illustrated in Figure 2.8.

2.14.6 Emerging Paradigms

Research and development is progressing on a number of novel interface techniques that may begin to enter mainstream use in the near future. Examples include Zooming User Interfaces (ZUIs) [16], attentive interfaces [117] and recognition-based interfaces (the contemporary name for “noncommand interfaces” [83] proposed by Nielsen). Those new interaction paradigms that follow the “direct manipulation” approach could be considered compatible with most accessibility techniques developed to-date; such techniques centre around rendering discrete interface objects, outputs and events in ways that are accessible to the user.

Recognition-based interfaces are, undoubtedly, becoming more popular [1] and they pose new accessibility challenges, as there are no discrete events: input and output are continuous and context is key to interpreting user intent. Though improving the accessibility of such interfaces is a laudable goal, it is beyond the scope of this thesis, where the focus will be on developing platform- and domain-agnostic techniques to improve the current and medium-term situation (WIMP and post-WIMP direct manipulation interfaces).

In terms of distributing ATs to people and easing the burden on platform vendors in doing so, the GPII project represents an ideal vehicle for the processes
developed here, particularly the “match maker” component\textsuperscript{35} and the fact that, as with Sus-IT, profiles are delivered using semantic web formats.

## 2.15 Goals and Scope

It is acknowledged that a comprehensive solution to the problems above would be both social and technical in nature, with social needs driving any technical innovation deemed necessary. This work aims to provide key pieces of the technical infrastructure that may be used as part of such a solution. Sus-IT aims to address a number of these social concerns. With respect to the technical goals identified, Figure 2.9 is a copy of Figure 1.1 for convenience. It illustrates the goals of both Sus-IT and this thesis, which are given below.

### 2.15.1 Approach

The present system is designed to assist people with minor-to-moderate impairments. As such, it would not be possible for such a system to account, in detail, for all possible situations in which a user may require assistance, nor the reasons

\[\text{http://gpii.net/components}\]
for the requirement, nor too for it to be aware of all possible solutions to accessibility problems. In fact, as has been asserted, the most profound adaptations are those that users may make without even thinking about it (such as using reading glasses or moving to a different position for better lighting). Therefore the present work aims to develop a reasoning process which can react to users’ needs and augments the assistance of which people are already aware. By expressing the reasoning problem at a suitably portable, yet expressive, level, it should be possible to highlight and make helpful recommendations for additional forms of assistance, of which the user may be unaware.

It should be emphasised that, as the severity of impairments in the target audience is not great, it should be possible for the system to glean much information from the user and—when appropriate—require feedback from the user in order to tune the reasoning process. In light of this, the theory and design work of this thesis adheres to the following key principles.

**Problem-centred.** The experiences and actions of the person using the system are vital to the success of the reasoning process. Computation should be directed towards what is computable efficiently; the user should be treated as an expert on matters relating to her or his feelings and preferences.

**Focussing effort where it is most needed.** The ultimate goal is to match user to appropriate adaptations. Exemplary systems already exist to model specific impairments and carry out adjustments for the user. The reasoning process must provide an effective way to “home in” on the adaptations which may be required and then instigate and defer to that help as quickly as possible.

**Formally-inclined, but not exact.** As above, the reasoning process operates
at too broad a level to be able to precisely model all relevant actors and phenomena. However, its design and testing have been inspired by a range of formal approaches, which are of great use in improving and assessing its efficacy.

**Human terms.** As discussed in sections 2.7 to 2.9, interactive system design techniques that emphasise the importance of the human player (such as user-centred design and design for all) are of increasing prominence. This work seeks to make human, as opposed to device, capabilities the basis for reasoning.

In order to achieve these goals, best practice\(^{36}\) has been adopted. Examples include: adopting a layered design for each component of the system (in order to separate different types of reasoning, or different levels of complexity) and developing as many components as possible in isolation (in order to keep them as simple and focussed as possible).

### 2.15.2 Selected Technical Goals

The goals to be addressed by this thesis specifically provide the core reasoning components of the system. This work does not, for example, seek to implement a full abstract UI system, as this is a problem addressed extensively by prior research. The contribution of this thesis is the process for reasoning with and about capabilities and adaptations and is embodied by the following goals.

- **Goal 2**, Accessibility barrier causes—though only the DUET capabilities will be considered, in order to keep the techniques developed generic and therefore more readily usable to developers.

- **Goal 3**, Simultaneous accessibility barriers

- **Goal 5**, Compensating for accessibility barriers

- **Goal 6**, Consistency of Interaction with Assistive Technology (support for this in the reasoning process; UI development to be carried out by Sus-IT)

- **Goal 8**, Multiple optimality (basic reasoning support; further development for abstract applications in Sus-IT)

- **Goal 1**, Collaboration

\(^{36}\)such as that of the “UNIX Philosophy” [http://en.wikipedia.org/w/index.php?title=Unix_philosophy&oldid=483805933](http://en.wikipedia.org/w/index.php?title=Unix_philosophy&oldid=483805933)
The use of human, rather than device-specific, capabilities as a portable yet still accurate barometer for adaptivity is also a contribution of this work; it is developed in chapters 3 and 4. The above goals support improvement in the following broad areas.

- Reasoning about accessibility, in terms of DUET constraints—thus mapping users with minor-to-moderate impairments to appropriate accessibility adaptations.

- Enabling “good-enough” bootstrapping on new devices or in new situations, based on users’ known capability (and underlying) preferences.

- Operating as passively as possible, allowing users to benefit either from improved adaptability or more active adaptivity, at their choice.

- Supporting accessible ubiquity and collaboration (particularly with respect to output rendering on shared devices).

- Providing a unified method for communicating to the adaptivity system; minimising interaction with ATs and maximising interaction with the application’s interface—both for end-users and, equally, platform vendors and developers.

As stated in subsections 2.11.7 and 2.12, one of the problems with current ATs and accessibility techniques developed by academia is that they have not been adopted by industry and mainstream products. Ultimately, the proof that the techniques developed in this work are of use would be such mainstream adoption. However, the timescale for this is likely to be large and there are limited resources available to accelerate this. For this reason, the following steps were taken to assess the validity of the methods developed.

**Technical testing.** Metrics were be developed to ensure that the techniques for adaptivity that are employed may be tested to demonstrate that they perform well in all anticipated situations.

**Sus-IT** is making use of the techniques developed—on behalf of both end-users and developers—and will report on the outcomes of longitudinal studies and work with developers in due course.

In an effort to keep the techniques developed as generic as possible—and to avoid the need for deep domain-specific knowledge, or a comprehensive knowledge base—the focus is on data-driven applications. It is anticipated that these restrictions will allow greater focus on the core problems (discussed in more detail in the
next chapter) and still yield results that should be transferable to more general application areas in future.

2.15.3 Sus-IT’s Wider Goals

For completeness, the list of additional goals that the parent work-package of Sus-IT is investigating follows.

- Application of adaptations (as directed by this thesis’ reasoning system) to legacy systems as well as abstract UIs. A framework for integrating existing legacy ATs and for affording capability-based atomic adaptations to web content are currently in testing.

- Developing and demonstrating adaptations for abstract UIs.

- Developing and demonstrating adaptations for legacy applications.

- Longitudinal testing of the system, based on the artefacts developed (this is at proof-of-concept stage at the time of writing [101]).

The next chapter discusses several approaches, operating over the short-, medium- and long-term, that have been taken or proposed to address the goals specified above.
Chapter 3

Approaches

This chapter discusses a number of potential approaches to solve the problems posed in the literature review, in the context of the literature and active research projects. Section 3.2 begins a discussion of the technical requirements for the chosen approach and the pre-requisite theory for addressing them, which is then compared at a high level to established practices and contemporary research.

As discussed in section 2.15, a comprehensive solution must be primarily social and part-technical. However, the rest of this work, and therefore this chapter, concentrates on the technical aspects.

3.1 Notable Approaches

This section highlights some historical and contemporary work that addresses at least some of the goals set out in section 2.15.

3.1.1 Technical

The general-purpose programming language “K” found extensive use in the financial sector, partly due to its very high performance (which, in turn, was due to most K programs being able to fit into the CPU’s cache). A key feature of this language was its simple, dynamic, data-driven approach to GUI programming, which allowed variables to be rendered differently, according to their type, with no extra work from the developer. For example: a one-dimensional array is rendered as a list box by default, but the developer may easily request a different representation, as shown in Figures 3.1 and 3.2.

This approach is in great contrast to contemporary software development tools; whilst it is possible in most languages to code the interface, it is almost always preferred to use a graphical UI design tool. In many ways, this is most appropriate
as UI design is an aesthetic task. However, the key idea from K that has not survived to the present was that the language would manage the rendering of the data in the most appropriate way (with minor hints from the developer). This was highly-suited to the highly data-driven applications for which K was designed.

### 3.1.2 Domain-specific vs. General

Many successful solutions to problems of adaptivity and accessibility have been focussed on particular tasks or particular domains [91]. These have been very successful in their own domains and demonstrate how well particular types of adaptation can serve users, if offered at the appropriate times and with the appropriate parameters. However, even the systems with user models are often couched in relatively high-level and domain-specific terms and are particularly aimed at content—rather than interface—adaptation, reducing their general applicability with respect to the goals of this work.

Other work has been interface-focused and, consequently, is considerably more general [40]. The example of SUPPLE is particularly relevant and, conceptually, goes a long way towards solving the goals set out in subsection 2.15.2. Considering each widget of the interface enables deep assessment of suitability to the given user and device. However, this approach of decomposing the problem into lower-level primitives is not applied to the user’s capabilities (the focus is on inferring and learning the user’s preferences in terms of widget usage only, via training both off-line and at run-time). This is natural, as the system was not designed to reason
CHAPTER 3. APPROACHES

Table 3.1: Comparison of approaches discussed.

<table>
<thead>
<tr>
<th>Name</th>
<th>Term</th>
<th>Acc.</th>
<th>Leg.</th>
<th>Data</th>
<th>M-m</th>
<th>Ubiq.</th>
<th>Rec.</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>K</td>
<td>Past</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>Obsolete</td>
</tr>
<tr>
<td>SUPPLE</td>
<td>Current</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>Modified</td>
</tr>
<tr>
<td>GADEA</td>
<td>Current</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>Current</td>
</tr>
<tr>
<td>(This work)</td>
<td>Medium</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>Modified</td>
</tr>
<tr>
<td>(Future)</td>
<td>Long</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>New</td>
</tr>
</tbody>
</table>

Fields: Acc. = the technique is geared towards accessibility; Leg. = it supports legacy applications; Data = the UI is data-driven; M-m = it is multi-modal; Ubiq. = it supports ubiquitous UI delivery; Rec. = it supports recognition-based UIs; Tools = nature of development style. Symbols: • = yes; ◦ = no.

3.1.3 Anticipated Long-term Developments

Given current paradigm shifts towards mobile and multitouch-based interfaces, the long-term popularity of recognition-based interfaces seems assured—as well as the gradual decline, at least from most users’ perspectives, of legacy “desktop” applications. These changes will necessitate a shift in development practices and environments and so present an ideal opportunity to build improved dynamic accessibility support into emerging development tools.

The move from current technologies to this future, however, will take time. Several barriers remain, such as recognition-based interfaces’ requirement for domain-specific knowledge and the lack of a structured and universal reasoning and knowledge storage framework, such as that promised by semantic web technologies. As discussed previously, it is beyond the scope of this work to tackle all of these challenges. For reasons of pragmatism, the techniques developed here aim to remain domain-agnostic. This does, however, present the opportunity to develop techniques that: (a) are applicable for the development of new devices and software today (with, perhaps, a moderate change in development practices) and (b) may provide useful knowledge, data and experience for future work. The nature of the main approaches discussed so far is summarised in Table 3.1.

The following section establishes the basis of the Capability-based Adaptive Accessibility Reasoning (CAAR) approach.

3.2 Theory Pre-requisites

Before any meaningful reasoning regarding capabilities, accessibility, adaptivity and interface distribution can be carried out, several key questions must be ad-
Figure 3.3: Example channels (visual, audio, keyboard) for a sighted user (left) and a blind user (right).

dressed, as follows.

1. What is the frame of reference for the system?

2. What is a capability and how might capability be measured?

3. How may capabilities, interfaces, adaptations and accessibility be represented in a machine-readable and machine-processable way and how might one capability affect others?

The goal of the theory used and developed is to express the problem at a sufficiently low level that these questions can be addressed in a computationally-efficient manner—but no lower, or applications built on top of an implemented system could be faced with having to implement some parts of the system themselves.

Whilst the areas of adaptation initiative and user interaction are beyond the domain of this thesis, they are key parts of Sus-IT and are discussed in subsection 3.5.3.

3.2.1 Frame of Reference: Conveying Information

The central focus of the work is to convey the interface to users in the most accessible way—i.e. maximising the amount of information they are able to process and presenting the information in a way they can perceive.

A system such as this, focussed on the transfer of information, often in situations where errors are likely to be introduced, or bandwidth constrained, clearly has strong relations to Shannon’s information theory [97]. Shannon modelled the transfer of information from transmitter to receiver over a potentially error-prone channel. Such a system is depicted in Figure 3.4 and has inspired the approach taken here.
3.2.2 Defining and Measuring Capabilities

Clearly the units of the quantities being processed (i.e. capabilities, constraints, accessibility barriers) must be compatible for the reasoning process to produce meaningful output. There follow two examples to demonstrate this.

Subsection 2.5.3 identified some basic situations in which accessibility barriers may exist. It is sometimes technically quite easy to detect a barrier. If the width of a wheelchair and the width of a corridor is known, then it is possible to determine the clearance between the two and reach a decision as to whether the wheelchair user should be directed down the corridor. This is because the two quantities are expressed in the same units and there may be guidelines that stipulate the recommended amount of clearance. Figure 3.5 expresses this situation. Naturally in reality, at least one of these quantities may be unknown.

Section 2.8 discussed some ways in which the decision as to whether an accessibility barrier is present, or affects the user, may be blurred. It is certainly not always a binary decision, or binary decision with some margin, as above. In fact it is perhaps more helpful if accessibility barriers are not considered as decisions; rather gaps in the capability requirements of the interface and the available DUET capabilities, as discussed in subsection 2.5.3.

Consider the example of speakers producing output for the user. The speakers will have a variable ability to produce sounds of certain frequencies. The ideal would be a uniform response, regardless of input frequency but this is not practically achievable. Figure 3.6 shows an example plot for a typical speaker.

We can compare this to a human’s ability to hear certain frequencies. Fig-
Figure 3.6: Frequency response—speakers (black line).

Figure 3.7: Frequency response—humans (overlaying that of the speakers). The signal output by a computer system must be within the perceptual range of the user (the solid and dashed blue lines represent different typical ranges).

Figure 3.8: Frequency response—environmental limitations (which, in this case, cut out a whole section of available frequency spectrum).

Figure 3.7 shows the speaker’s frequency response (solid black line) above along with two additional plots: the hearing of an average young human (solid blue line) and that of an older human (dotted blue line). From this plot, we can see that older users are less likely to be able to perceive or perhaps distinguish between higher frequencies.

Environmental factors are likely to play a significant part in the capacity of the channels between device and user. If the user is working in a noisy environment, entire portions of the spectrum may be inaudible to them. This would limit the range of frequencies that the computer system should output, as shown in Figure 3.8. The pertinent points of this example are as follows.

- The combined DUET capabilities applicable to a particular channel must
be considered before decisions on rendering in that channel should be made. The channel is the combination of user, device and environment at a particular time.

- The capabilities are expressed in terms of human perceptual capabilities. The motivation for this is discussed in the following section.

3.2.3 Representing Capabilities, Barriers and Adaptations

The question of what a capability is has been answered, albeit in a high-level manner, but this information is not useful unless it is coupled with a way to relate capabilities to each other. Modifications to how information is presented in one channel will likely affect other channels. Likewise, device-specific settings are likely to affect each other (e.g. increasing text size decreases available screen-space).

Contemporary work has sought to address these issues by making use of such standards as Composite Capability/Preference Profiles (CC/PP), User Agent Profiles (UAProf) and SNAPI [102]. However, these are somewhat device-specific and cannot relate device functions directly to human capability requirements for using those functions. As touched upon in subsection 2.8.2, there exists an internationally-recognised standard for classifying human capabilities—WHO’s ICF [124]. Existing work by Billi uses the ICF to assist with modelling human-computer interaction scenarios [19]. The approach taken by Billi is to extend the existing classification in two ways, as follows.

**Extending the human skills classification** via an appendix to include more fine-grained details such as the user’s performance in clicking, dragging and moving the mouse.

**Adding a device functionality appendix** in a similar vein to the extra human capabilities, which categorises device functions such as screen resolution, colour depth, whether joystick input is supported and whether voice input is supported.

It is argued that, although this approach could help improve modelling of interaction with—and therefore hopefully the accessibility of—current devices, it is *not the correct approach for the medium- and long-term*. This is because it is device-dependent (the example given was a capability that the “user can use the mouse wheel”). A more sustainable approach would be to express the additional classifications in terms of *human* functionality (e.g. “level of dexterity in fingers”), as (a) human capability classifications change incredibly slowly over
time\textsuperscript{1} and (b) expressing the capability requirements of device functions in terms of human capabilities allows the same classification to support new devices as they are developed (thus meeting the “human terms” goal from subsection 2.15.1).

Expressing as much data as possible in terms of human capability requirements has other significant benefits—it allows a relationship between adaptations to be defined. For example, the use of larger text may have a positive effect on both visual acuity and self-confidence\textsuperscript{2}.

### 3.3 The Capability-based Adaptive Accessibility Reasoning Approach

The goals identified in section 2.15 intend to support improvement in three broad areas, repeated below.

- **Reasoning about accessibility, in terms of DUET constraints**—thus mapping users with minor-to-moderate impairments to appropriate accessibility adaptations.
- **Enabling “good-enough” bootstrapping on new devices or in new situations**, based on users’ known capability (and underlying) preferences.
- **Operating as passively as possible**, allowing users to benefit either from improved adaptability or more active adaptivity, at their choice.
- **Supporting accessible ubiquity and collaboration** (particularly with respect to output rendering on shared devices).
- **Providing a unified method for communicating to the adaptivity system; minimising interaction with ATs and maximising interaction with the application’s interface**—both for end-users and, equally, platform vendors and developers.

In support of these goals, the following two methods are proposed, both of which require moderate changes in development tools and practices, albeit changes that are asserted to be in line with the current progression of technology.

#### 3.3.1 Standards

Existing domain standards such as the ICF \[124\] and technical standards such as RDF and XML should be used as the basis for the work, in order to benefit

\textsuperscript{1}For the purposes of this work, the gamut of human capabilities may be considered static.

\textsuperscript{2}… though cognitive modelling is out of the scope of this work.
from the effort, rigour and interoperability work that has been invested in those standards.

### 3.3.2 Deconstruct the Interface

There are many examples of where an integral consideration of accessibility can improve a product for many people; from universal design (section 2.7) to electronic systems where ATs are provided as part of the system (Table 2.3). However, there still are the problems of undiscoverability (subsection 2.11.1) and the static nature of contemporary commercial AT (subsection 2.11.4).

One next logical step to including an AT with a software product\(^3\) may well be to completely integrate it into that product. Instead of providing a separate program (process) to re-render GUI widgets in alternative forms, why not give the basic interface libraries the ability to render widgets in multiple modalities and formats, according to the needs of the user(s) at a given time and over the course of time?

A “molecular” approach to user interfaces would treat an interface not as a single artefact, mutable only as a whole or in large chunks, but a collection of more basic parts—the widgets, or data input/outputs—which could be distributed to an appropriate device, and rendered as required to optimise accessibility for the user over time. This would allow more fine-grained “adaptations” (now just alternative renderings) to be performed on the parts of an interface that a user is having difficulty with—and for portions of the interface to be moved to other devices as and when appropriate.

Continuing with the idea of breaking the interface down into component parts, we may wish to consider the interface of the entire system—i.e. the interface components presented by the OE and any running applications—as one global tree of interface components. These may be rendered in many different combinations, according to the configuration that suits the user’s needs at a given time.

### 3.3.3 Domain-agnostic Decision-making

As previously detailed, this work aims to provide a bridge between future techniques that may involve comprehensive reasoning based on domain-specific knowledge and are able to employ recognition-based interfaces and the current paradigms of direct-manipulation. The techniques developed here are targeted at data-driven applications as their interaction semantics are likely to change more slowly over time. Should the techniques be successful, they will provide a migration path to

---

\(^3\)as demonstrated by Mac OS X and VoiceOver
future, recognition-based development methodologies. In the context of a system considering DUET capabilities, a number of decisions have to be made, as follows.

- What adaptations to, or renderings of, data are required?
- When must they be made, to what extent, in what area of the interface and for how long?
- How can an abstract interface be best targeted at a specific set of devices, users, environments?
- How can sensor data (such as light and sound levels) and user feedback (passive, or active, over the course of time) be taken into consideration?

It must be borne in mind that as well as the decision-making process, accurate input to, and a way to use the output from, this process are needed. These would be analogous to the plug-ins in the Sus-IT architecture that monitor the environment and user’s actions, as well as those that affect adaptations (the deconstructed ATs themselves).

### 3.3.4 Layers of Reasoning

Figure 3.9 shows the overall architecture of the reasoning process. This is separated into layers. From bottom to top the principle is to progressively discount solutions to collaboration or adaptation problems that are determined infeasible, in order to reduce the search space at the next level upwards in the chain. The reasoning processes become more sophisticated at each successive level.

### 3.3.5 Reasoning in Action: Decision Trees

As discussed in section 2.15, there may be multiple different ways to render an interface in an accessible manner to the user. The notion of a set of widgets with many different possible renderings invites the notion of a large decision tree, which could result in many permutations of these widgets and renderings. By considering these choices in the interface generation process, a unified method can be developed for users to choose between different possibilities—allowing the system to learn their preferences—and to indicate dissatisfaction with the—or any part of the—interface.

Such a large tree would (and could) not be explored in its entirety, as will be demonstrated. In light of supporting potentially fluctuating capabilities, it is perhaps more helpful to think of such a tree as the possible sets of adaptations which could be applied in a given “problem situation” that has been detected or
Figure 3.9: Conceptual layers of reasoning. Reading order is bottom-up, although execution order would not be linear, as higher layers will feed back to lower layers in order to improve the suitability of solutions generated. Areas below the dotted line are the primary focus of CAAR and the thesis.
Figure 3.10: Possible adaptations example. The underlying nature of adaptations is discussed in future chapters. The null adaptation is needed in order to ensure proper exploration of the decision tree.

Figure 3.11: Adaptation sets decision tree example. From the root (a detected or declared problem), the available adaptations are explored. Each leaf node represents a set of proposed adaptations. The null adaptation is needed to ensure that intermediate solutions are also enumerated.
declared (e.g. the user has low visual acuity, or environmental factors are effectively causing impairment, as discussed in subsection 3.2.2). Possible adaptations are given in Figure 3.10 and possible uses of these are depicted by Figure 3.11.

3.3.6 Run-time Components

Though this thesis is concerned with the theory and technical approach towards molecular user interfaces, as opposed to developing applications, it is necessary to consider the components that would be required by a run-time supporting such interfaces. As discussed in subsection 3.3.3, this involves two major tasks: reasoning about the interface (which can be further subdivided) and those activities that surround the process—receiving input data and affecting chosen renderings. Figure 3.12 provides a visual overview of how these processes are related; discussion on how this has been adapted into a usable system for Sus-IT can be found in Appendix B.

3.4 Relation to Thesis Goals

Here the potential effects and benefits of implementing a molecular interface are discussed.

Multiple Optimality. As discussed, decision trees, in conjunction with elicited user preferences, may be used by the reasoning process to evaluate different solutions to certain accessibility problems.

Unified Interfaces. As the interface of the OS and all running applications forms part of the global interface tree, they can be split up and rendered as appropriate. This, for example, affords the ability to render information of a low-priority (such as the progress of a long-running or background task) differently from parts of the interface that the user is interacting with regularly, or that have high priority.

Distributed Operation. A natural outcome of the interface being considered as a global tree and the anticipation of run-time manipulation of the tree is that parts of it may be rendered on different devices and even moved between devices.

Co-operative and Distributed Co-operative Tasks. Further, parts of the interface tree may be delegated to other people as well as devices. The issues of conflicting access needs and adaptations are of the utmost importance
Figure 3.12: Architecture of proposed approach—run-time components.
here; the system must be designed to reason about the interaction between two (or more) sets of DUET capabilities at once.

The Temporal Dimension. The main challenge to the system is the complications introduced by considering the temporal dimension at design-time. As discussed in the previous chapter, many adaptive systems are able to react to changing needs—even predicted changes—but not to changes in device, environment and collaboration over time, and perhaps not with the assumption that capabilities vary in patterns over multiple timescales. Chapter 7 is concerned with this topic.

As discussed, there is little data on this variation in capabilities over time. Though Sus-IT is working to provide such data, an adaptive system is required to monitor and validate this data. Therefore the techniques developed in this work do not make any assumptions about human capabilities; they are designed to work within a wide range of different scenarios. Chapter 8 provides more discussion on this subject.

3.5 Application: the Sus-IT Project

As discussed, the CAAR process forms one of the contributions of the wider Sus-IT project, with the goals of matching people to the appropriate ATs and adaptations for both legacy and more adaptable applications. Whilst the rest of the thesis discusses the potential of CAAR in the ideal setting of abstract applications, the process is also applicable in legacy settings (the “adaptations” available are simply coarser, due to the monolithic nature of ATs and relatively immutable nature of most legacy applications). This section describes how the CAAR approach is being applied by Sus-IT to legacy systems.

Contemporary ATs are, as discussed, expensive and static in nature—and are developed under the assumption that they are to be used in isolation from other ATs. Let it be assumed that, due to the potentially large market for adaptive systems, developers would be amenable to using a framework for adaptivity if doing so required little (and preferably no) effort on their part.

To this end, it is proposed that the operating system should include a lightweight library that is transparently linked to all applications (in much the same way that MSAA or AT-SPI are now) to allow them to communicate with individual adaptations. These adaptations may be interface renderers, allowing widgets to be presented in a range of modalities, or input handlers that may perform operations such as key debouncing.

There already exist a wide range of preference and machine-learning systems
that could be used and enhanced to track user preferences over time and across different applications [40, 34]. Given a suitable user model and problem detection framework (see below), difficulties that the user experiences could be tracked in a similar way, to allow future decisions on adaptations to be more accurately calculated.

Each individual adaptation “plug-in” would be very simple; capable of only one style of rendering—such as text-to-speech—and would have the ability to take parameters controlling the adaptation. The system would need to determine which adaptations should be brought in at any given time; this could be achieved by way of a controller process (provided with the OS/adaptivity library), which would respond to predicted or detected problems the user faces, as well as environmental constraints. Multiple adaptations would be usable concurrently on a given interface (or parts of it) so, for example, the user whose eyes tire when reading a large amount may have certain types of content spoken aloud but toolbars and menus simply enlarged.

As much effort as possible should be made to keep preferences system-wide; the need for this was discussed in relation to IBM WAT in subsection 2.12.2 above.

Figures 3.13, 3.14 and 3.15 show an example of how such an approach may be structured when implemented and how it compares to current systems, both with and without AT. It should be noted that the lightweight adaptation framework could easily be developed separately from and later installed into the operating system (this would be required during the proof-of-concept stage), though it would ideally be included in the operating system in the long-term.
3.5.1 Potential Pitfalls, Criticisms and Justifications

As discussed in section 2.3, a number of criticisms have been directed towards adaptive approaches in the past. This section revisits some of these criticisms in the context of the adaptive accessibility (as opposed to interface) approach proposed.

Specificity—flexibility trade-off. Adaptive interfaces that are highly-domain specific may be able to afford the user some very useful adaptations. However, those that seek to be more generic may fail to do so, as the problem becomes computationally intractable and, thus, the task model becomes relatively more rudimentary. Myers et. al. refer to a problem of threshold and ceiling—the ideal systems being low-threshold and high-ceiling, which is very hard to achieve generally (though some techniques for allowing the user to discover more advanced features of an interface, such as the “trapdoor” method, hold some promise) [79].

People do not like change. Accessibility adaptations are aimed more at enabling users to use a technology than refining the usability of it. For this reason, an adaptive system for accessibility need not make such high-resolution predictions about the user as a system dealing with small-scale usability enhancements. Therefore, it is argued that accessibility changes made via the vehicle of adaptivity will be more successful than those changes previously made by adaptive systems in the name of usability. In addition, the changes made will most likely be either very gradual (in line with capability decline) or more marked but welcome (e.g. increasing font sizes for users whose eyes are known to tire each afternoon).

Use users, not models Sometimes models are required because it is not possible to communicate with the user to ascertain their needs. Fortunately, in the context of adaptive UIs for those with minor to moderate disabilit-
ies, it is quite likely that communication with the user will be possible to corroborate or refine the predictions the reasoning process makes about the changes they may require at a given point. This data can be used to assist the machine in making future decisions and hopefully allow it to “back off” from communicating with the user at a burdening frequency.

Two other sources of data that can be passively used by the system include: changes the user makes to the system (such as global colour settings) and environmental changes (in, for example, light or sound levels, which can be detected using sensors either built in to or attached as peripherals to the computer).

**User model inaccuracy.** In fact this problem may be considerably smaller for an accessibility-oriented (rather than usability-oriented) system due to the generally coarser nature of accessibility barriers. It is possible to make measurements of the environment of a user passively by means of sensors available on modern computers (particularly portables, where environmental changes are likely to have a greater impact on accessibility factors). An adaptivity framework would also have access to historic data on usage and how the environmental constraints (e.g. light and sound levels) have affected this. Available adaptations and past usage data for them are also available, as well as records of when the user has countermanded a setting that the system changed automatically.

**One size does not fit all.** Taking the previous point further; the user model can be extremely light and largely exist as a process that captures and analyses the user’s own actions (with some reasoning in order to mitigate the problem of users unaware of the assistance available). Effectively the reasoner becomes a reflection of the user’s own historic behaviour.

**Bootstrapping** It is important to keep in mind that a suitable set of initial questions given to the user (in an accessible way) may be very helpful in determining which adaptations (and more specific detection routines) may be required. Perhaps when the social challenge of encouraging people to expect adaptivity features has been overcome, this will cease to be as much of a problem because (a) users will accept some bootstrapping and (b) user profile data portability standards will inevitably be developed to minimise the need for bootstrapping when users move between systems.

Sus-IT’s approach to this problem has been to develop “mini-games” to assess capabilities of users in an engaging way. These are based on standard
psychomotor tests and were developed with advice from a specialist work-
package within the project—but developed in such a way that some accur-
acy was traded for a considerably less clinical, more engaging, approach.
Examples are given in Figure 3.16. (The viability of these games is being
evaluated by the wider project.)

It should be borne in mind that the suggested approach does not constitute yet
another new standard that will need to be evangelised and followed. It is simply
an approach that could be used to provide a framework for adaptivity—and would
stand a much greater chance of success if it relied on existing open standards. For
this reason, protocols such as the W3C’s capabilities and profiles (CC/PP)\textsuperscript{4}
could be used to express device—and possibly user and environmental—constraints.

Further: it is highly important that the data collected by such systems are
portable so that users can receive appropriate adaptations on any system they
use. Here again open standards such as RDF \textsuperscript{5} could be of use.

3.5.2 Infrastructure for Adaptation Discovery and
Installation

It is not proposed that the operating system must include all conceivable adaptations—
rather that problems are detected and adaptation plug-ins be downloaded when
required (possibly including a purchasing stage). Adaptations could be advertised
via a range of directory services, possibly hosted by the operating system vendor
and funded by revenue from adaptation purchases.\textsuperscript{5}

3.5.3 Interaction Workflow

The design goals for the interaction process, being developed for Sus-IT are as
follows.

Minimalism. Adaptation interactions must not dominate the use of an ICT.

Simplicity. In order to achieve minimalism, simplicity of the workflow is desired
(few clicks, distractions).

Consistency. Interaction with and control of adaptations must be carried out by
the same means.

Transparency. The user should be able to interrogate the system to ascertain
what data it is using and how.

\textsuperscript{4}http://www.w3.org/TR/CCPP-struct-vocab/
\textsuperscript{5}This approach have succeeded elsewhere: http://www.macrumors.com/2008/12/05/apple-advertises-300-million-apps-downloaded-over-10000-apps/.
Figure 3.16: Example capability engagement games, developed by Yunqiu Li and the author (top) and Kirsty Smith (bottom). The top “games” present a simple direct choice to the user; the bottom one involves performing an action (double-clicking to move the cart across the screen).
This section describes the possible interactions between the user and user-visible parts of the system. Figure 3.17 illustrates an example dialogue that may be used to allow the user to review adaptation suggestions.

- The user or the system may apply adaptations (but in this section the focus is on the interaction from the user to the system).

- At any time the user may...
  - Carry out an adaptation (as normal).
  - Request help (an adaptation).
  - Provide feedback on an applied adaptation (either by the user or the system), in the form of acceptance or rejection. (For more complex feedback, a request for help may be used.)

- If the user requests help, there are multiple stages (all encapsulated in one dialogue; see Figure 3.17).
  - The current “top n” adaptations are proposed. These may be activated (or, for advanced users, edited by removing adaptations or altering parameters).
  - There is also a choice to dismiss the dialogue entirely.
  - The user may indicate that none of the proposed sets of adaptations are appropriate and ask for more to be generated.

- After an adaptation, the user is given the opportunity to accept or reject the adaptation (or, as ever, request help again). Feedback may be supplied by unobtrusive on-screen icons, keyboard shortcuts or dedicated hardware (similarly for help requests).

3.5.4 “Help Button” and Adaptation Suggestions

Part of the workflow described above involves allowing people to request help if they feel this is necessary. A universal way in which to provide this facility was designed: a “help button”, a mock-up of which is depicted in Figure 3.18. When activated, the user would optionally be able to direct the request for help to a specific application (or, in the case of abstract applications, part thereof). Following this, the system would suggest appropriate adaptations, using an interface such as that depicted in Figure 3.17. Subsection 9.2.2 discusses the matter of how adaptations may be presented to users.
Advanced users would be given the option to reconfigure the adaptations proposed in a given set, as is shown in this figure—however these options would be hidden by default so that novice users are not overwhelmed. The manner in which previews of the adaptations are affected would vary, but actually applying the adaptations to the system is argued to be the best approach, where possible.

Figure 3.18: “Help button” mock-up and Sus-IT software version (which is always accessible on-screen).
3.5.5 Testing the Concept

The concept of an adaptive system; its potential to help people and possible social and ethical pitfalls have been explored with older people and other target users. Details of “sandpit” activities, which gleaned useful insight into people’s feelings towards such systems are detailed elsewhere [101]. The “sandpits” used a combination of interactive theatre and “blue-sky” product design sessions with users (seeded with examples created by the investigation team) in order to elicit participants’ feelings and views.

Of particular interest from these early findings was that older people who had previously used personal computers seemed a little skeptical about some adaptive features, citing privacy concerns—would the local optician be told if a depredation in vision were detected? Also of concern was the trend towards dedicated appliances over using custom software on a general purpose computer.

However, people who did not identify themselves as “computer users” reacted positively to the idea of appliances that might help them communicate with others more easily via adaptation. People in this group seemed more comfortable with the system “learning” about them, in exchange for the extra assistance that could be provided.

The designs of longitudinal and short-term assessments of the CAAR process, including user interaction and capability tracking, are given in chapter 9.

3.6 Summary

Various historic and contemporary approaches to the problems posed have been discussed. Based on the various successes and limitations of these approaches, the nature of the theory developed in this work—CAAR—has been outlined and a number of aspects of this theory that must be created have been identified. Further, the central position of this work (the core reasoning framework) within work-package four of the Sus-IT project has been highlighted, in order to further explain the scope of the work and how it is being longitudinally tested and deployed in the real world.
Chapter 4

Fundamental Reasoning Actors

This chapter introduces and defines the actors in the reasoning process and their pertinent properties. These are then used to develop each stage of the reasoning process proper in the following chapters.

4.1 Proposed Usability/Accessibility Indicator

A definition of usability and accessibility that is not dependent on technical objectives, but real-world usage by a population or user, is proposed as follows. The definition is designed to be simple and form the basis of a rough indicator of an artefact’s usability and accessibility. This indicator could be used to direct the application of more rigorous and involved analysis where warranted.

- If someone can use an artefact, at any particular time, it is usable. The number of people that can use it at any particular time is denoted by $u$.

- If someone cannot use the artefact, at a particular time, it is inaccessible. The number of people who cannot use the artefact at a particular time is denoted by $x$.

The artefact and method of determining whether the artefact is usable should remain as constant as practically possible. Other factors are expected to vary. Again, the goal is to determine an indication of real-world usability and accessibility, so although more data would likely lead to a more accurate result, that result would still be just an indication.

An example: if an artefact is usable by ten people, but not usable by two people, then it may be considered “mostly usable” but also “partly inaccessible”. The above definitions account for the fact that one user may find an artefact both usable and inaccessible (e.g. with and without reading glasses).
The notion of “at any time” is important: systems could be assessed over time, in different usage (i.e. DUET) scenarios. Further, the usability and accessibility of a system may vary over time for each user—in which case $u$ would denote the number of times that the user was assessed to be able to use the system and $x$ the number of times the system was found to be inaccessible to the same user.

The level of confidence in the measurements increases with the value of $u + x$ and the level of confidence in the general usability of an artefact increases with the ratio of $u$ to $x$ and value of $u + x$.

### 4.1.1 Rate and Scope

A “hit rate” metric is defined as follows. This tracks, over time, how usable and accessible an artefact is for a given user or population of users (and would lie between 0 and 1 inclusive).

$$r = \frac{u}{u + x} \quad (4.1)$$

The hit rate may be recorded with respect to a given artefact and one of the following.

**A population of users at an instant in time.** This would indicate how many currently find a given system to be usable, or inaccessible ($r$).

**A population of users over time.** This would record how the usability of a system changes over time for the population, given trends in usage patterns and devices ($\frac{dr}{dt}$).

**A user over time.** $r$ would represent an moving average over the period of time ($\frac{dr}{dt}$ could also be recorded, as above).

Each “usage” of the artefact by an individual user contributes to the hit rate, as above. The scope of what constitutes an individual “usage” may vary, depending on the resolution desired, as follows.

**An interaction session** such as the use of a wordprocessor to write a letter (very coarse assessment).

**An interaction task** such as saving the document, as part of the letter-writing process (less coarse).

**Interaction with a sub-artefact** such as the “Save...” dialogue box (a finer resolution of assessment).
An artefact may be seen as being composed of sub-artefacts. Continuing the example of a wordprocessor, each of the following, progressively smaller, sub-artefacts may be seen as artefacts in their own right and have a usability and accessibility result associated with them.

- The entire wordprocessor. Recording $U/X$ decisions would result in a very coarse indication.

- The save dialogue box, which allows the user to carry out certain tasks and has its own particular internal workflow.

- Individual widgets within the dialogue may present usability or accessibility barriers and may be considered as artefacts—though it is most likely that barriers would result from the placement of widgets relative to each other, or cognitive mismatches between the required workflow and that expected by the user.

The level at which the hit rate indicator is assessed—which part of the artefact is having its usability and accessibility measured and over what timescale—would depend greatly on the intended goal of the assessment. For example: testing a replacement “Save…” dialogue may benefit from assessment of both the overall task of saving a document (in order to consider the broad interaction of the dialogue with the rest of the process), as well as monitoring of sub-tasks within the dialogue itself (in order to get a higher-resolution view of the efficacy of the replacement).

It is accepted that not all tasks will be atomically usable or inaccessible. However they may only be recorded as such using this metric (of course it is perfectly possible for qualitative or categorical data to be recorded alongside these measurements). The reason for this simplicity is that as the hit rate is only a rough indication, any more complex analysis based on it would likely yield unreliable or misleading results.

4.1.2 Features of the Indicator

The indicator gives the following benefits.

- It is population-based, therefore reflects the current state of real-world usage, rather than compliance with given objectives. One aim would be for “crowdsourcing” of accessibility data.\footnote{Perhaps more automated, but sharing the philosophy of \url{http://www.fixtheweb.net/} (accessed 26/02/2012).}
• It tracks usage over time, for a user and a population and reflects the usability and accessibility of the popular mode of access of a service/artefact.

• It is portable across different technologies—in the methodology sense; it is not claimed that meaningful comparison of results could be achieved as they are most likely too high-level.

• If it were adopted it would allow developers to “home in” and invest more effort in improving areas that are in most need. There is a prevalence of social media sharing and “like” buttons on web pages and within web applications; a similar rapid feedback approach may be successful.

Of course it has the disadvantages that it cannot be measured exactly, nor does it indicate why an artefact or part thereof may be inaccessible. However, the reasoning processes developed in these chapters aim to complement the indicator by providing such information.

4.2 Relating the User, Adaptations, Devices and Capabilities

Though these terms will be defined more formally below, it is important to have a high-level idea of how they are related. Figure 4.1 illustrates the key relationships between all actors in the system. Salient points indicted by Figure 4.1 and others of relevance are as follows.

• Users have capabilities. The applications and content (modified perhaps by adaptations) require capabilities of the user; the environment can bring further capability constraints. These capabilities fall into the established major channels: the four main modalities, plus volume of information and temporal constraints (e.g. how fast a task may be completed; a lack of time on the part of the user/situation).

• “Within” the standard modalities, there are more precise capabilities, such as those laid out by the ICF.

• It is far easier to note which channel/modality a given capability resides in than assess or determine its exact nature (in ICF-like terms).

For example: certain types of adaptation belie changes in capabilities afforded within certain modalities; text enlargement may imply a problem with visual clarity. However, others may be applied for a wider range of reasons.
Figure 4.1: Adaptation Reasoning Entities. It is important to note that each device renders or accepts input via only one channel and that applications may be rendered across multiple devices.
• Interaction between the user, on the one hand, and applications, information (content) and adaptations, on the other, occurs at the device level. Applications, or parts of them, are rendered to one or more devices (e.g. monitor, braille display). The application may also render content.

Some adaptations may be applicable to content, but these will most likely be executed in conjunction with the application with which the content is associated, rather than on the content directly.

• Perhaps the most profound types of adaptations that the user may employ are undetectable by the system. These would include a change of glasses or alterations to the environment that are undetectable by the system [95]. This is not to denigrate the importance of adaptations under the system’s control, but it is important to recognise that the system should not expect to be aware of the full picture.

• Though the notion of adaptations as separate processes to applications is reinforced here, this is not the ideal; as with Gajos’ work, in future it is hoped that adaptations will be considered as an integral part of the system, becoming more akin to the parameters with which widgets, applications and devices are instantiated.

4.2.1 Focusing on Information Transfer

Again, the task of the system is to transfer as much information as possible to the user (which inherently implies doing so in a way that is perceivable and understandable by the user). Figure 4.2 is a simplified version of Figure 4.1 that highlights the relationships between the actors above (simplified) and the transfer of information to the user. From Figure 4.2 the following additional properties of the relationships between entities can be observed (the diagram would be similar when taking into account the transfer of information from the user to the application via the device, so is not presented here).

• The capability requirements on the user may be formed from those of the: Information; Application(s) and Device(s) being accessed/in use.

(This is because although the user only directly receives information from the device, the information still passes through all of the other stages.)

• These requirements are further affected by: temporal factors (such as time of day; user tiredness or device power or connectivity); the Environment and any Adaptations affecting any of the above elements.
The information transfer time is directly affected by all of the actors that the user is affected by (including the user’s own level of capability) as well as the indirect actors/entities, as above.

- Adaptations applied to the Environment can indirectly affect devices and users.

- Adaptations to devices that change the nature of temporal factors (e.g. lowering screen brightness or disabling network access to increase battery life) may in turn affect the user and, thus, the transfer of information.

These concepts will be used in chapter 6.

### 4.2.2 Classifying Capabilities and Components

From discussion in the previous chapters, there is a substantial body of work regarding the classification of capabilities upon which CAAR may be based. The ICF in particular has a focus on human capabilities, is internationally recognised and work has already been carried out that demonstrated its potential for improving interactions’ accessibility [19]. Clearly an electronic standard for describing and classifying human capabilities is required in order for the capabilities of users—and the capability requirements of interfaces and devices—to be processed and manipulated by a computer.

Further, as discussed, any descriptions of interface and device capability requirements should be based on the standard for describing human, as opposed to machine, capabilities. This implies a layered structure which is depicted by Figure 4.3.
For any practical implementation of a capability reasoning system, certain standards must be developed in order to ensure a common frame of reference. This is also true of adaptations and other elements of the computing system to be introduced shortly. In practice, it should be possible to describe the components of a person—those organs or structures that afford capabilities—in a machine-readable way. Such a standard would enable users and their profiles to indicate preferences as to which components be used in certain situations (such as being left- or right-handed).

However, CAAR requires only that there is such a classification, providing: (1) a way to uniquely identify each possible capability and to link this to (2) a given modality and (3) the component(s) of the body that may afford that capability. For this reason, the reader may assume that such a classification exists and provides the requisite information. Those interested may read Appendix C to learn how the classification was developed in order to make it suitable for practical use by Sus-IT.

### 4.3 Information and Device Characteristics

Figure 4.4 shows the overall process of the transfer of stored data in a machine to being information understood by a human (and vice-versa). This section covers most of these stages (in both input and output directions), detailing the relevant characteristics of the information and mapping each of these to the human and the device viewpoint.

One striking consequence of the communication between human and computer (or other humans) is that, no matter how complex or abstract any given piece of information may be in one’s mind (e.g. a map; music; text), it must be encoded in one of the three main interaction modalities in order to be transmitted between the parties involved.

Output devices convert computer data types to impulses that a human sens-
Figure 4.4: The transfer of stored data into understood information. Various stages of this process are shown, from the computer (bottom) to user (top). Data must be transferred between people and devices via the visual, auditory or motor modalities.
CHAPTER 4. FUNDAMENTAL REASONING ACTORS

Table 4.1: Summary of the dimension and temporal nature of static information, streams and control events.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>◦</td>
<td>◦</td>
<td>Control event</td>
<td>Function key press</td>
</tr>
<tr>
<td>0</td>
<td>•</td>
<td>•</td>
<td>Control signal</td>
<td>Mouse cursor position</td>
</tr>
<tr>
<td>1</td>
<td>◦</td>
<td>•</td>
<td>Static stream</td>
<td>Text (prose); music</td>
</tr>
<tr>
<td>1</td>
<td>•</td>
<td>•</td>
<td>Temporal stream</td>
<td>Collaboratively-edited text</td>
</tr>
<tr>
<td>2</td>
<td>◦</td>
<td>◦</td>
<td>Static plane</td>
<td>Image</td>
</tr>
<tr>
<td>2</td>
<td>•</td>
<td>◦</td>
<td>Temporal plane</td>
<td>Film</td>
</tr>
</tbody>
</table>

**Fields:** Dim. = dimension of data; Temporal? = Does the information—not just one’s perception of it—change over time?; Stream? = is the information part of a series? **Symbols:** • = yes; ◦ = no.

The sensory organ can understand. The organ sends this data to the brain, where it is interpreted and expanded into information. Conversely: the brain decomposes information (as a computer cannot understand this) into data of low-level type(s). It then encodes this data in terms of the required actuations (e.g. key presses; vocalisations) to convey that data to the input device, where it is converted into low-level computer data for storage.

It is important to be aware that the manner in which the mind expands data—for example: to form an impression of the real-world scene represented by a two-dimensional photograph—is heavily based on the experiences of the person involved. Depending on these experiences, significantly more information may be gleaned from the photograph than may be measured by size of its storage file in a computer. In fact, this is the most important point about this process: the brain expands data to form information.

It is beyond the scope of this work to model this expansion, but it is important to note that the purpose of any computer-aided communication system is to most readily facilitate the user’s understanding of what is being communicated. For different types of ultimate information (e.g. 3D scene; timetable), different adaptations to the source data may be preferred to improve its accessibility. One real-world example of this is given by different people’s preference for presentational style of learning materials—visual, audio or kinaesthetic.

Table 4.1 summarises the general types of data and control signals that are often transferred, and relates this to their dimension and modality. Two important points arise from this table, as follows.

- Some data types may be thought of as a stream—though this does not imply that the data themselves change over time; rather that they may be rendered over time.
The rendering time of some data is not necessarily obvious from its dimension and temporal nature (and is influenced by many cognitive factors that are out of the scope of this work).

Finally, it is important to continue to note that any transmitted information will likely be expanded significantly in peoples’ minds—e.g. a photograph becomes a real three-dimensional scene; a recorded symphony becomes a real orchestral experience. These technical sections are concerned with the data transmitted, which is of significantly lower size and dimension, but the end purpose of the transmission must be borne in mind at all times.

### 4.3.1 Modalities and Data Types

Table 4.2 lists the fundamental modalities considered here for input and output. Note that cognition is not used directly for input/output; rather it is used by a human to process any given data to yield information, or to prepare the appropriate components to produce output.

Table 4.3 gives a set of fundamental data types and their symbols. In the table, the “M” column denotes the “natural modality” of the data—the manner in which it is most readily represented. The “Axes” column denotes how many degrees of freedom an individual datum may possess—a stream of mouse position updates, for example, is one-dimensional and temporal in nature, but each position update represents a delta to a coordinate with two degrees of freedom.

Regarding haptic input/output, the major focus will be placed on the one-dimensional form (under which Braille output will be classified, as each Braille character may be interpreted instantaneously by a fingertip) rather than the two-dimensional form (e.g. a tactile image or texture).

The first group of data types is abstract: these types can not be input or output directly; instead they must be encoded as data in another modality via one of the concrete types in the lower groups.

Some of the groupings of types within Table 4.3 require explanation. The selection of symbols allows for bitmap and vector formats for images and video to
Table 4.3: Fundamental data types (machine perspective). The first group is of abstract types, which cannot be directly transmitted. Note that human “data types” such as “colour” and “shape” would be encoded in the machine as one of the following.

<table>
<thead>
<tr>
<th>Dim.</th>
<th>Modality</th>
<th>Type</th>
<th>Axes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0/1</td>
<td>Cognitive</td>
<td>Number</td>
<td>$n \in \mathbb{N}$</td>
</tr>
<tr>
<td>0/1</td>
<td>Cognitive</td>
<td>Character or text</td>
<td>1</td>
</tr>
<tr>
<td>0/1</td>
<td>Cognitive</td>
<td>Boolean</td>
<td>1</td>
</tr>
<tr>
<td>0/1</td>
<td>Cognitive</td>
<td>Action</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Auditory</td>
<td>Recorded Speech</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Auditory</td>
<td>Music</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Auditory</td>
<td>Other sound</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Visual</td>
<td>Bitmap image</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Visual</td>
<td>Vector image</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Visual</td>
<td>Bitmap video</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Visual</td>
<td>Vector animation</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Motor</td>
<td>Haptic (stream)</td>
<td>1–3</td>
</tr>
<tr>
<td>2</td>
<td>Motor</td>
<td>Haptic (area)</td>
<td>1</td>
</tr>
</tbody>
</table>

**Fields:** Dim. = Dimension; Axes = degrees of freedom within the stream.

**Note:** Data of any type will take time to render and may also change over time.

be considered as just “images” or “video” instead of being more specific. This is because (a) the human sees both bitmap and vector images as simply “images” (though may prefer different adaptations depending on the image type—for maps versus photographs, for example) and (b) it would be possible for the computer to determine the format of any given image.\(^2\) The types of recorded sound are not grouped, however, because (a) they would be perceived very differently by the human and (b) it would be non-trivial and possibly intractable for a computer to make the determination as to the type of a given sound file.

For these reasons, the burden of recognition is placed on the computer when dealing with vector(bitmap) detection and the human when classifying sounds. This is another example of problem-centred computing and illustrates the assignment of problem-solving sub-tasks to the party that would most readily be able to accomplish them. Its relevance to the adaptation reasoning will become apparent later.

\(^2\)Though problems in the real world may be caused by the inappropriate use of bitmap formats for vector images, these are not considered here.
4.3.2 Multiplexing

Consider the example of presenting a public information announcement via the mass media. Over radio, the announcement must be repeated every so-often, in the hope that it reaches the public. Over television, a banner may be permanently shown as part of the broadcast video stream. These are classic examples of serial/parallel in terms of information transmission, or time/space sharing, in terms of an OS managing peripheral devices.

Figure 4.5 shows how multiplexing is achieved on a one-dimensional device—by allocating part of each time period to the different sources of information (normal programming and announcement in this example)—and on a two-dimensional output device—by sharing the available bandwidth between the sources of information, so that both may be displayed concurrently and constantly (avoiding problems of the audience missing the announcement).

4.3.3 Organisation of Devices

Figure 4.6 relates devices (of progressively more specialised nature) to the types of data channels that are used to communicate with users. It is important to note that the non-interactive portions (e.g. RAM, CPU) of a computing device are ignored as they are not in direct communication with users. Further, devices are arranged in a tree structure. Four terms are defined to refer to the structure of devices within the tree.

**Parent devices** contain other devices and are denoted $D$. They form tree or sub-tree roots (e.g. as shown in the figure, the desktop computer logically
Figure 4.6: Devices and information.
Transmission devices (T-devices) are contained by parent devices (e.g. a microphone built into a computer; the buttons on a mouse). They are the only devices that render information for, or receive input from, their users. These are denoted $d$ and form leaf nodes.

Soft devices are special T-devices in that they do not have physical form themselves; they must be rendered to the user via a different T-device (e.g. a “soft keyboard” being rendered on a display screen). No soft devices are shown in the figure but, as with T-devices, they would also constitute leaf nodes only.

Drivers are supplementary processing stages (either post-input or pre-output) that the computer can execute in order to convert one form of data into another (e.g. TTS).

Parent devices may form only the root and mid-level nodes in the tree; T-devices may form only leaf nodes.

4.4 Applications and Content

The final entity that must be specified before interesting problems can be solved is the application with which the user wishes to interact.

The work of Gajos provides an exemplary manner in which to model and render abstract UIs, taking device constraints and users’ preferences—and some capabilities—into account and is detailed extensively elsewhere [42]. This work builds upon that foundation, but also to support existing legacy applications and adaptations, in order to render them as accessible as possible despite having relatively inflexible interfaces. Therefore, the standard adopted is based on that of Gajos’ “SUPPLE” system, but with some relaxation as to the required level of detail of the interface trees, as well as some consideration of capabilities.

It should also be made clear that this work does not seek to duplicate any earlier work on abstract UIs and therefore defers to previous projects such as SUPPLE to provide the algorithmic contributions and validation of such an approach. This work seeks to show how the abstract approach may be used to facilitate and improve the provision of adaptive accessibility in the context of collaborations and fluctuating human capabilities.
4.4.1 Interface Tree Summary

As with Gajos’ work, applications’ interfaces are modelled as trees of abstract widgets. The job of the UIMS is to match the abstract widgets to the appropriate concrete widgets for the current user and device. Gajos’ work takes into consideration a user’s individual usage patterns and preferences with respect to widget types. For example: when selecting the appropriate widget for integer input, SUPPLE++ will consider the following.

- Device constraints (e.g. screen size).
- Widget usage frequency.
- Widget usage nature (i.e. concrete widget choices may reflect the amount by which values in widgets are usually adjusted).
- Other user preferences, vision and motor capabilities (e.g. preferred font or widget size).

Several basic types of widgets are defined and these broadly correspond to the abstract data types given in Table 4.3—i.e. the interface specification gives not only the structural arrangement of widgets, but also the widgets’ purpose in supplying information to the program from the user (and, in the case of this work, vice-versa).

This work makes one extension to Gajos’ model, which then allows for some further simplification: an additional node type, representing an “application part”—an abstract “chunk” of an application (such as a “Save…” dialogue or document window). The purpose of this addition is to make it possible to specify an application at varying resolutions. This extension also allows some simplification: application parts also replace Gajos’ “group” and “interface” (root) abstract widget types. The UIMS would have to decide whether to render an application part as a window in its own right or perhaps as a tab group within a larger window. This could easily be done based on either a simple heuristic (e.g. how close the node is to the root of the application tree) or using Gajos’ own methods for reasoning about widget prominence based on usage patterns (e.g. more commonly-used parts may be exposed as part of the main window).\footnote{Earlier criticism of “unpredictable” adaptive systems such as personalised menus—and its relevance here—is noted. Gajos’ solution to this was to designate a specific area of an application as the “adaptive area” thus preventing some parts of the application from changing unexpectedly, yet still affording the user some shortcuts and flexibility through adaptivity. In this case, a robust heuristic, or developer mark-up, could fulfil the same requirement.}

Ideally an application would have a totally abstract UI and be modelled down to each abstract widget, so that the adaptive UIMS can select the most appropriate
devices, concrete widgets and any required or appropriate adaptations for each portion of the interface. However, legacy applications that cannot be modified at such a fine resolution also need to be supported. Figure 4.7 gives small examples of each type of modelling—abstract and legacy, respectively (the legacy application’s specification in fact does provide some widgets explicitly, so it is more of a hybrid abstract-legacy specification).

4.4.2 Applications, Data and Capabilities

Applications are made up of application parts and abstract widgets (primarily for input—though they will almost certainly have to be rendered on an output device so as to be usable for the user) or content items (primarily for output—though some, e.g. wordprocessor documents, will be editable). The root part represents the application; mid-level parts represent sub-applications (which may be rendered as separate windows in a GUI) or groups of abstract widgets and content (which may be rendered as, for example, frames or tab page groups in GUls). Abstract widgets and content items are referred to broadly as “abstract entities” for brevity.

Each part \( (P) \) is a group of other parts and may also contain abstract entities \((e)\). Each abstract entity is represented by a data descriptor which defines the following.

- A descriptor to denote the type of data being conveyed (Table 4.3).
- Links to the capabilities required in order to process the data. These would not need to include those of the T-device as they would already be known. (A standard library of capabilities based on datatype and ultimate rendering parameters would be used in practice; only additions pertaining to this particular content would be required.)
- A link to the data to be rendered—such as a text or image file on a disk, or a variable being processed by the program as it runs.
- An optional ordered list of alternative content descriptors and links, to be used in the event that the preferred content is inaccessible.

Aside from the hierarchical structure another important property of any application or part thereof is the ability for the abstract widgets or content items to be navigable in a linear fashion (analogous to the “tab-order” of GUI widgets).
Figure 4.7: Example of: a fully abstract Gajos-style UI specification (top); the same model expressed in CAAR and a specification of a legacy application that is only partially adaptable (bottom).
4.5 Summary

The most fundamental elements of the reasoning process have been defined. In particular, the following aspects are of note.

- Data- (and soon, capability-) oriented ways of specifying all of the entities involved, from devices to applications.

- The ability to specify abstract and legacy applications (and hybrids) to varying degrees of resolution, using a unified model.

In the following chapters, the reasoning process will now be developed by the next three chapters, as depicted in Figure 4.8.

Chapter 5 establishes techniques for affording accessible collaborations. Building on the level of capability reasoning developed so far, this allows for a worked example in which a person with a severe impairment wishes to work with a person without impairment.

Chapter 6 introduces the notion of an adaptation and its effects on information transfer. This allows the capability reasoning process to be refined and developed in order to cope with less clear-cut situations, involving minor-to-moderate impairments.

Chapter 7 discusses some challenges involved in maintaining the effectiveness of the system over time and proposes two main ways in which the work could be extended to facilitate this.

Finally, a technical analysis, designs for various types of interaction and longitudinal testing, to be carried out by Sus-IT, and the conclusions and further work chapters follow.
Chapter 5

Collaboration as Synchronisation

Based on the elements introduced in the previous chapter, basic reasoning regarding collaboration and very coarse adaptations is introduced. The nature of a collaboration problem is described and techniques are developed for modelling and solving such problems.

5.1 Synchronisation

It is now possible to begin some basic reasoning. Synchronisation, as termed by this work, is the process by which the actions of one user, device or application within a system are represented through another device, or to another user or application. This section details the three types of synchronisation—which are essentially the same technique, applied in slightly differing domains: matching cursors to input and output devices; matching local collaborating users and devices and matching remote collaborating users, devices and applications.

5.1.1 Relating Input, Focus and Devices

Users’ interaction with a device is not in isolation: the effects of input provided via one device are often made apparent through another. A classic example of this is the visible mouse cursor present in all WIMP GUIs, which allows the user to relate their manipulation of the mouse device to the (virtual) manipulation of on-screen objects. However: (a) there are many more cases where input and output must be synchronised and (b) due to DUET constraints, the optimal pairing may not always be possible (or obvious), so an alternative must be sought. A generic technique for pairing devices must therefore be developed.

In order to develop such a technique, the relevant entities must be identified and described. The relevant entities here, in addition to the input supplied by and
output sent to the user, are as follows.

The “focus” is the concept of where input from the user will be directed within the system when received by a device. It is important that the user is able to easily (a) determine where the focus currently is and (b) move it to another part of the system. Focus is often indicated in GUIs by visible highlighting of the currently-selected widget.

The focus is effectively the “focal point” of the synchronisation.

Cursors of different types, which represent the synchronisation between input and output devices and are rendered via the output device. Cursors can often be used to change the focus, particularly in GUIs.

The devices themselves. These are either physically connected or otherwise in communication with each other.

The applications or parts thereof, which receive input from some devices and are rendered as output via others.

Table 5.1 lists a range of properties of four UI paradigms and the cursors found within them. Some key observations regarding this data follow.

- The OS filters all input, so system-wide gestures input via any device may be captured before they would reach applications.

- Gestures may apply to individual widgets, but these are not considered here.

- “Activates?” refers to the cursor triggering an action on an application part, for example a widget such as a toolbar button is activated by a finger tap in a multitouch GUI, or a mouse-click in a WIMP system.

- In multitouch systems, the cursor corresponds directly to the user’s haptic input (tapping or dragging) and only exists whilst that input is taking place. This means that a tap indicates both position and the intent of the user to activate the widget at that position. In WIMP systems, however, position is supplied via an indirect mouse pointer and is separate from actions such as a mouse click.

- “Axes” refers to the same concept as in subsection 4.3.1 and is explained in more detail below.

- The position of any of the cursors may be represented by a number; either linear for the tab-order, or with two axes in the case of the GUI pointer.

Finally, Table 5.2 gives definitions of the three main cursor types. Subsequent sections define the generalised synchronisation method for devices.
Table 5.1: Properties of cursors in different UI systems. Comparison of how the cursors are controlled and the types of control/information they allow the user to impart to the system. Cursor properties and control methods vary with UI styles.

<table>
<thead>
<tr>
<th>Concepts</th>
<th>WIMP GUI</th>
<th>CLI</th>
<th>Multitouch GUI</th>
<th>Multitouch ZUI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Window</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤/○</td>
<td>○</td>
</tr>
<tr>
<td>Widget</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>Document</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Position cursor (*e.g.* mouse pointer)

<table>
<thead>
<tr>
<th>Presence</th>
<th>Permanent</th>
<th>–</th>
<th>Transient</th>
<th>Transient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focuses?</td>
<td>Optional</td>
<td>–</td>
<td>Always</td>
<td>Always</td>
</tr>
<tr>
<td>Activates?</td>
<td>Never</td>
<td>–</td>
<td>Always</td>
<td>Always</td>
</tr>
<tr>
<td>Scope</td>
<td>System</td>
<td>–</td>
<td>System</td>
<td>System</td>
</tr>
<tr>
<td>Max num.</td>
<td>1</td>
<td>–</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Axes</td>
<td>2</td>
<td>–</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Movement</td>
<td>Free</td>
<td>–</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>Render mod.</td>
<td>V</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Input mod.</td>
<td>M</td>
<td>–</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>

Direct widget navigation (*e.g.* keyboard “tab order”)

<table>
<thead>
<tr>
<th>Presence</th>
<th>Permanent</th>
<th>–</th>
<th>Permanent</th>
<th>Transient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focuses?</td>
<td>Always</td>
<td>–</td>
<td>Always</td>
<td>Always</td>
</tr>
<tr>
<td>Scope</td>
<td>Application</td>
<td>–</td>
<td>Application</td>
<td>Application</td>
</tr>
<tr>
<td>Max num.</td>
<td>1</td>
<td>–</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Axes</td>
<td>1</td>
<td>–</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Movement</td>
<td>Only widgets</td>
<td>–</td>
<td>Only widgets</td>
<td>Only widgets</td>
</tr>
<tr>
<td>Render mod.</td>
<td>V</td>
<td>–</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>Input mod.</td>
<td>M</td>
<td>–</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>

Editing cursor (*e.g.* “I-beam”)

<table>
<thead>
<tr>
<th>Presence</th>
<th>Transient</th>
<th>Permanent</th>
<th>Transient</th>
<th>Transient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focuses?</td>
<td>Always</td>
<td>–</td>
<td>Always</td>
<td>Always</td>
</tr>
<tr>
<td>Scope</td>
<td>Widget</td>
<td>–</td>
<td>System</td>
<td>System</td>
</tr>
<tr>
<td>Max num.</td>
<td>1</td>
<td>–</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Axes</td>
<td>1/2</td>
<td>–</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>Movement</td>
<td>Free</td>
<td>–</td>
<td>Free</td>
<td>Free</td>
</tr>
<tr>
<td>Render mod.</td>
<td>V</td>
<td>V</td>
<td>V</td>
<td>V</td>
</tr>
<tr>
<td>Input mod.</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>

**Fields:** Focuses? = does the cursor inherently give focus?; Scope = area within which the cursor may be found; Max num. = permitted in the system simultaneously; Axes = degrees of freedom; Render mod. = natural rendering modality (without ATs); Input mod. = natural input modality.

**Symbols:** ⬤ = yes; ○ = no; – = not applicable.

**Notes:** (1) via application; (2) inherent (documents are a first-order element of the UI); (3) some window managers auto-focus windows under the pointer; (4) extra input required (e.g. mouse-click); (5) rarely used; (6) widget must be editable; (7) linear (though some systems offer the shortcut of using the up and down arrow keys) or using position cursor to move the editing cursor.
Table 5.2: Formal properties of cursors.

<table>
<thead>
<tr>
<th>Entity</th>
<th>Dim.</th>
<th>Temporal</th>
<th>Type</th>
<th>Axes</th>
<th>Capabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position cursor</td>
<td>0</td>
<td>●</td>
<td>$T_N$</td>
<td>2</td>
<td>$C_{\text{position,activation}}$</td>
</tr>
<tr>
<td>Direct widget navigation</td>
<td>0</td>
<td>●</td>
<td>$T_N$</td>
<td>1</td>
<td>$C_{\text{ordering}}$</td>
</tr>
<tr>
<td>Editing cursor</td>
<td>0</td>
<td>●</td>
<td>$T_N$</td>
<td>2</td>
<td>$C_{\text{position,editing}}$</td>
</tr>
</tbody>
</table>

Symbols: ● = yes; o = no; data type information given in Table 4.1.

5.2 Example Scenario

Figure 5.1 depicts an application that has been shared between two users. The details of the scenario are as follows.

User 1 can interact in all concrete modalities given in Table 4.2 ($V, A, M$).

User 2 can interact in only audio and motor modalities ($A, M$).

There is a shared computer, as well as one for each user. Each computer has several T-devices: Screen, Speakers, Keyboard, Mouse.

5.2.1 Cursor Assignments

Devices need to be assigned to allow users to manipulate the focus. Table 5.2 lists two groups of cursors: those for focusing on a widget and the editing cursor which enables the user to input data. When assigning these to devices, the following must be taken into consideration.

Cursor rendering must be assigned to the T-device that renders the associated widget. By default this is the highest-dimension T-device that meets the requirements of the entity/cursor (i.e. a 2D T-device for most users).

For User 1 this would be a screen. As this is a 2D device, it can accommodate the positional cursor, which affords random access to the UI, so this would be the primary cursor for User 1.

For User 2 this would be an audio or perhaps haptic device. As these devices are linear in nature, the “tab-order” direct widget navigation cursor is selected. This also means that User 1 has the ability to move between widgets outside of this linear order and this could disrupt User 2.

Cursor input is assigned in a similar fashion.

User 1 will most likely use the mouse as the editing cursor, as most textboxes, is 2D in nature.
Figure 5.1: Example entity–T-device assignments for a given application (dashed line shows abstract widget linear order). Each entity (abstract widget) is assigned to a set of T-devices for input and output, for each user. This figure represents one solution out of many possible solutions for two users and a range of devices.
User 2 will in this case have to use this 2D cursor as it is the only one available. Two devices are present that can control it: a mouse and the keyboard (arrow keys). The mouse is effectively linked to a “soft” T-device that is rendered on a screen. This is inaccessible to User 2, so the keyboard, which is entirely physical in nature and thus accessible, is selected.

The next section (and chapter) generalises these processes.

5.3 Defining Synchronisation

There are three basic instances of synchronisation between the members of a group of users—all of which are requisites for collaboration.¹

Entities. Can each user be presented with either the same or equivalent concrete renditions of the abstract widgets and content items from the application?

Focus. Can each user be made aware of the area of the application that has another user’s attention?

Updates. Can each user be made aware of updates to the state of the application, made by any user?

5.3.1 Coherence

The coherence of each instance of synchronisation listed above may be one of the following.

Identical. All users share the same instance of the entity (i.e. it is on a device shared by everyone).

Strong. All users have the same representation of the entity: they are all viewing a copy of the same content, rendered in the same modality.²

Weak. Some users have differing views of the content, either due to: (a) the content having been substituted for an alternative, so that it can be perceived (e.g. “alt-text” for images) or (b) automatic translation into other modalities (e.g. TTS as opposed to visual text rendering).

Very weak. At least one user cannot access the content at all.

¹Cognition to a certain level is of course also required; this is peripheral to this work but will be touched upon in the following chapters

²Adaptations, discussed in the following chapter, may be applied in order to help the users perceive the content, but if the content itself is unaltered—i.e. still rendered in the same modality and has not been substituted or translated—then the synchronisation is strong.
Naturally “strong” synchronisation is preferred, so that all users share the same frame of reference and, to the maximum extent possible, rendering time.

### 5.3.2 Interference

The level of interference, likewise, is categorised as follows.

**Passive.** The users can perceive changes in focus, or updates, without having their rendering of or interaction with the application interrupted.

**Disruptive.** A user’s rendering of or interaction with the application is interrupted in order to reflect changes in focus or other updates.

Naturally, more successful synchronisation scenarios would be passive, so that users’ concentration is not broken unnecessarily. Figure 5.2 visually represents the process for determining the nature of synchronisation for one particular part/entity, being rendered for a group of users, either on one local device or multiple remote devices.

### 5.3.3 Optimal Content and Alternatives

As well as sharing the same representation of an abstract entity, users would ideally be presented with the representation that gives them most information. This, however, is separate from the level of coherence of the synchronisation, as summarised in Table 5.3.
Table 5.3: Examples of synchronisation properties (output). Two users wish to collaborate: given their capabilities to access particular modalities—and T-devices in use—the possible synchronisation types are shown.

<table>
<thead>
<tr>
<th>Entity Type</th>
<th>T-devices</th>
<th>Perceivable?</th>
<th>Coherence</th>
<th>Interference</th>
<th>Optimal?</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1: V, A, M; User 2: A, M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text</td>
<td>Screen</td>
<td>●</td>
<td>–</td>
<td>–</td>
<td>●</td>
</tr>
<tr>
<td>Text</td>
<td>Speakers</td>
<td>○</td>
<td>Strong</td>
<td>Disruptive</td>
<td>○</td>
</tr>
<tr>
<td>Text</td>
<td>Screen, Speakers</td>
<td>●</td>
<td>Weak</td>
<td>Disruptive</td>
<td>●</td>
</tr>
<tr>
<td>Image</td>
<td>Screen</td>
<td>●</td>
<td>○</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Image</td>
<td>Speakers</td>
<td>○</td>
<td>Strong</td>
<td>Disruptive</td>
<td>○</td>
</tr>
<tr>
<td>Image</td>
<td>Screen, Speakers</td>
<td>●</td>
<td>Weak</td>
<td>Disruptive</td>
<td>●</td>
</tr>
</tbody>
</table>

Symbols: ● = yes; ○ = no; – = not applicable.

The table insinuates that it is possible to assign an image to be output via speakers. This is possible if alternative content has been provided—if it had not then the coherence would be very weak. Text may be assigned to Screen or Speakers by means of lossless conversion—though the result is limited by the dimension of the rendering device.

5.3.4 Replacements as Coarse Adaptations

The example above has already highlighted the most basic type of adaptation, in which one device would be replaced with another to satisfy capability constraints. The same technique may be applied to the abstract entities of an application, also. The classic example of this is substituting images with alternative text for people who cannot see well enough to perceive the image.

Such replacements can be thought of as content substitutions. The designer of an interface or author of some content may provide alternatives (which would be modelled as alternative data descriptors) for a given abstract entity. Therefore, the most coarse adaptation that could be made is making a substitution for such an alternative. This is not always necessary or desirable (a more refined approach is developed in the following chapter), but does constitute one of a number of available courses of action open to the reasoning process.
5.4 Ideal Collaboration Solutions

A process is needed to match users to both T-devices and abstract entities. This section describes how an “ideal” such process would work—ideal in the sense that this is most similar to contemporary approaches such as SUPPLE, but has been extended to multi-user, multi-T-device problems.

The outcome of the process is a set of assignments that match users to both T-devices and application parts or abstract entities, of the following form (referred to as a “solution”). The users and devices must be present in a given environment $E$; the collaboration is to occur within application $P$. An example, in which the application from section 4.4 is being used by two users, is shown in Figure 5.1, which represents one solution out of the many possible.

\[
S(U, D, P) := (entity1_assignments, \\
entity2_assignments, \\
\vdots \\
entity2_assignments)
\]  
(5.1)

where

\[
entity\_assignment := (shared\_output\_tdevices, shared\_input\_tdevices, \\
user1\_output\_tdevices, user1\_input\_tdevices, \\
\vdots \\
userN\_output\_tdevices, userN\_input\_tdevices)
\]

and

\[
U \in U \text{ (users in environment)} \\
d \in D, \ D \in D \text{ (T-devices in parent devices in environment)} \\
e \in P, \ P \in P \text{ (entities in application part(s))}
\]

Naturally an optimal solution would maximise the amount of data that the user can perceive from (and transmit to) the system and would be consistent with user preferences. The concept of user preferences is introduced in chapters 6 and 7. In this example, the perception of data and the ability to transmit it are relatively clear, given the stark nature of the example scenario.

Application parts themselves are not the subject of assignments: as parts contain entities, they would need to be rendered in the same manner as the entities they contain. If the entities of a given part are split across multiple T-devices
(as may happen when a user is controlling one device using another), then a representation of the containing part would have to be rendered on each T-device; this means that the assignments for parts are implicit, given the assignments involving entities.

### 5.4.1 Search Space and Technique

The nature of one solution is detailed above, however there are many possible solutions for a given problem; the process that finds the optimal solution must assess a potentially vast number of candidates along the way. The simplest (and time-consuming) approach would be an exploration of all possible solutions for the given application, users and devices. This exploration itself takes the form of a tree, as partially depicted in Figure 5.3.

However, even for a small example, the search space can be vast, as in the following example (which involves a very small application).

\[
\begin{align*}
\text{num_users} &= 2 \\
\text{num_entities} &= 7 \\
\text{shared_output_tdevices} &= 2 \text{ (screen, speakers)} \\
\text{shared_input_tdevices} &= 2 \text{ (mouse, keyboard)} \\
\text{user1_output_tdevices} &= 2 \text{ (screen, speakers)} \\
\text{user1_input_tdevices} &= 2 \text{ (mouse, keyboard)} \\
\text{user2_output_tdevices} &= 1 \text{ (speakers)} \\
\text{user2_input_tdevices} &= 1 \text{ (keyboard)}
\end{align*}
\]

An individual entity assignment (as above) is of the following form.

\[
\text{entity_assignment} := (\text{shared\_output\_tdevices}, \text{shared\_input\_tdevices}, \\
\text{user1\_output\_tdevices}, \text{user1\_input\_tdevices}, \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \ldots \\
\text{userN\_output\_tdevices}, \text{userN\_input\_tdevices})
\]

Each set of devices is any combination—i.e. the powerset—of the available T-
Figure 5.3: Example of search tree traversal where only one device is assigned to each user for each entity. This is a virtual search space, composed of all possible solutions, which are constructed progressively from one level to the next. Each node of this tree corresponds to one possible partial or complete solution (set of assignments of the interface tree’s elements to users and T-devices). At each node, there are $\#\!\!_d^{|U|}$ branches (the number of possible assignments of devices amongst the users). The depth of the tree is $|P|$ (i.e. the number of entities). Cost analysis is carried out at each node, allowing many nodes to be pruned. A dotted line highlights one solution’s construction path (i.e. from the root to a leaf node).
devices, so the cardinalities involved, in this case, are as follows.

\[
\text{assignments} = |\mathcal{P}(\text{shared}_\text{output}_T\text{devices})| \\
\times |\mathcal{P}(\text{shared}_\text{input}_T\text{devices})| \\
\times |\mathcal{P}(\text{user}_1\text{output}_T\text{devices})| \\
\times |\mathcal{P}(\text{user}_1\text{input}_T\text{devices})| \\
\times |\mathcal{P}(\text{user}_2\text{output}_T\text{devices})| \\
\times |\mathcal{P}(\text{user}_2\text{input}_T\text{devices})| \\
= 2^2 \times 2^2 \times 2^2 \times 2 \times 2 \\
= 4 \times 4 \times 4 \times 2 \times 2 \\
= 4^5 \\
= 1,024
\]

Each entity may (non-exclusively) be allocated any of these assignments.

\[
\text{possible solutions} = \text{assignments}^{\text{entities}} \\
= 1,024 \times 10^7 \\
= 1.18 \times 10^{21}
\]

The upper maximum size of the search space is, therefore, calculated as follows.

\[
s = \left(2^i \times 2^o \times \prod_{U \in \mathcal{U}} (2^{i(U)} \times 2^{o(U)})\right)^{e} \tag{5.2}
\]

where

\[
i \text{ is the number of shared input T-devices.} \\
o \text{ is the number of shared output T-devices.} \\
\mathcal{U} \text{ is the set of users.} \\
i(U) \text{ is the number of input T-devices for user } U. \\
o(U) \text{ is the number of output T-devices for user } U. \\
e \text{ is the number of entities.}
\]

This is also the formula for the number of nodes at any non-root level of the virtual search tree shown in Figure 5.3, in which \(e\) would be replaced by the level number.

Due to the vast search space, a branch-and-bound approach [72] is used. This guarantees that the search will be complete (all possible solutions will either be evaluated or correctly discounted). In order to effectively employ branch-and-
bound searching (detailed in algorithm 5.1), the following measures are taken.

Algorithm 5.1 search($S, E$).

Require: Partial solution $S$; remaining entities $E$ as parameters and the following
global variables.

Sets $I$ of shared input T-devices, $O$ of shared output T-devices.
Set $U$ of users; functions $I(u)$ and $O(u)$ to return these sets for users.

$best\_costs \leftarrow \infty$; $S \leftarrow \emptyset$

Ensure: Solution $S$ is the lowest-cost scenario matching each entity $e$ of $E$ to
shared and/or user T-device assignments.

1: if $meets\_constraints(S) \neq T$ then
  2:     return \{Constraints not met; prune this sub-tree of potential solutions.\}
3: end if
4: if $estimated\_cost(S) \neq T$ then
  5:     return \{Best cost appears unbeatable with this solution; prune.\}
6: end if
7: if $is\_complete\_solution(S) = T$ then
  8:     \{Solution is accepted as the global best.\}
  9:     $best\_costs \leftarrow cost$
10:    $best\_solution \leftarrow S$
11: else
12:     \{Continue to enumerate all possible assignments, across all users.\}
13:     $e \leftarrow select\_unassigned\_entity(E)$
14:     $device\_sets\_list \leftarrow \emptyset$
15:     push($device\_sets\_list, P(order\_devices(I)))$
16:     push($device\_sets\_list, P(order\_devices(O)))$
17:     for $u \in U$ do
18:         push($device\_sets\_list, P(order\_devices(U(I))))$
19:         push($device\_sets\_list, P(order\_devices(U(O))))$
20:     end for
21:     for $device\_assignments\_set \in cross\_product(device\_sets\_list)$ do
22:         assign($e, device\_assignments\_set$)
23:         search($S, E$)
24:     end for
25: end if
26: return

A cost function, or series of cost factors that assess the suitability of any
given solution and assign a cost to it. The cost function would be composed
of a range of different objective measures of cost. Examples would include
a cost associated with how many parent devices are shared between users—in
a collaborative situation it is often desirable to maximise this. Costs
introduced later will include the predicted amount of information transfer
that a given concrete version of the interface may provide.
Cumulative cost prediction (an admissible heuristic). The cost functions must be capable of reliably predicting the minimum cost of a partially-built solution (in the case of a complete solution they would return the actual cost of that solution). If, as a solution is being built, its predicted cost is higher than a solution that has already been assessed, then the solution—and all of its descendants—can be discounted and need not be evaluated further.

Ordering of variables (entities) and values (T-devices). As all nodes are ultimately visited or reliably discounted, the order in which the nodes (partial solutions) are visited does not affect completeness. However, it has been shown that the order in which solutions are built can have a significant impact on the run-time of the search [42]. There are two aspects to the ordering: (1) the order in which unassigned abstract entities are selected and (2) the order in which the possible T-devices which form potential assignment sets are made.

The “minimum remaining values” ordering method, in which both aspects of ordering are carried out on a “most-constrained first” basis, was found to be particularly useful for problems of this type [42].

Constraints and constraint propagation. Constraints could be thought of as much “harder” cost factors, in that they provide a means to make a binary (accept/reject) decision regarding a particular solution (and its descendants) and are, therefore, more coarse than the cost factors introduced above. Typically constraints are used by designers of systems to assert that certain conditions should never arise; e.g. in abstract user interfaces, if it is considered imperative for all toolbar buttons to be rendered on the same T-device, then the designer may specify a constraint that any solution placing one or more toolbar buttons on a different device to the rest must be rejected.

Because they are very coarse, constraints can provide an efficient way to prune swathes of illegal solutions, but are typically used sparingly for precisely the same reason.

5.4.2 Constraints and Cost Factors

Constraints used for pruning swathes of the potential solutions tree are listed in Table 5.4. A number of cost factors apply to the process of finding an optimal set of devices and renderings, for both individual users and those wishing to work together. The desired goal for a factor—whether it should be minimised, maximised or is irrelevant—varies with the type of collaboration, e.g. local or remote working. The factors are described in Table 5.5. Two types of cost factor are
Table 5.4: Constraints affecting the collaboration and users.

<table>
<thead>
<tr>
<th>Name</th>
<th>Measuring</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoPartFragmentation</td>
<td>Do not split parts across parent devices.</td>
</tr>
<tr>
<td>AllEntitiesShared</td>
<td>All entities are assigned to shared devices.</td>
</tr>
<tr>
<td>NoUnreachableSharedEntities</td>
<td>Shared entities are perceivable by all users.</td>
</tr>
<tr>
<td>NoUserPersonalDevices</td>
<td>Do not assign to users’ personal devices.</td>
</tr>
</tbody>
</table>

Table 5.5: Cost factors affecting collaborations and users.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Measured quantity</th>
<th>Typical goals</th>
<th>Affected area of collaboration</th>
</tr>
</thead>
<tbody>
<tr>
<td>User group</td>
<td>$D_1 \cap D_n$</td>
<td>min, max or neither</td>
<td>Shared parent devices.</td>
</tr>
<tr>
<td>User group</td>
<td>$P_1 \cap P_n$</td>
<td>min, max or neither</td>
<td>Shared application parts.</td>
</tr>
<tr>
<td>User</td>
<td>Non-optimal</td>
<td>min</td>
<td>Avoids assignments of entities to T-devices with lower dimensions.</td>
</tr>
</tbody>
</table>

given in the table: those that apply only to collaboration situations (with more than one user) and those that apply to individual users, whether they are taking part in a collaboration or not.

5.5 Collaboration Scenarios and Cost Factors

A large range of different collaboration situations can be modelled based on the factors given above. This section details the modelling of both local and remote scenarios of different types.

5.5.1 Local Collaboration Variants

The three main types of collaboration that can occur locally are as follows. Each of the scenarios can be created by varying the goals for the factors given above.

**Local mirrored** collaboration is a situation in which all users wish to work on the same devices with the same application parts.

**Local ability-based** collaboration involves users sharing devices as much as possible but application parts are allocated according to users’ individual capabilities.

**Local delegated** working is a scenario whereby users wish to share as many devices as possible, but also focus on application parts specific to them.
Table 5.6: All possible types of collaboration, based on different cost factor goals.

<table>
<thead>
<tr>
<th>Type</th>
<th>Users nearby?</th>
<th>$D_1 \cap D_n$</th>
<th>$P_1 \cap P_n$</th>
<th>$I$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local mirrored</td>
<td>◦</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Local ability</td>
<td>◦</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Local delegated</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Near mirrored</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Near ability</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Near delegated</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Remote mirrored</td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Remote ability</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
<tr>
<td>Remote delegated</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>●</td>
</tr>
</tbody>
</table>

Fields: Users nearby? = are users in the same vicinity?; $D_1 \cap D_n$ = the set of shared parent devices across all users; $P_1 \cap P_n$ = the set of shared application parts across all users; $I$ = the amount of information for each user.

Symbols: • = yes; ○ = no; – = not applicable.

One further point to bear in mind is that, when users are working in the same environment (and regardless of whether they are collaborating), they may well be able to communicate via non-electronic (e.g. verbal) means.

5.5.2 Remote User and Application Synchronisation

The three variants on local collaboration scenarios may be mirrored in two further situations, in which: (1) users who are in the same vicinity, do not necessarily wish to work together but are able to communicate with each other using non-electronic means and (2) users who are in different environments and therefore may only communicate via electronic means. These extra sets of scenarios, sharing much in terms of factors with the ones discussed already, are given in Table 5.6.

5.5.3 Combination of Cost Factors

Algorithm 5.2 shows how the cost factors are combined. The traditional branch-and-bound approach is used and still one solution only is selected as the best, however the cost factors are considered as separate functions, not combined by methods such as weighted sum. This allows some cost factors to be “biggest-best” (in which case the comparison directions would be reversed; this is not shown in the algorithm for simplicity). This allows the presence of certain cost factors and their directions to be tuned.

It should be noted that this is not intended to approximate the Pareto frontier
Algorithm 5.2 \texttt{estimated\_cost}(S).

\textbf{Require:} Partial solution $S$.
\quad Vector $M$ of metric functions.
\quad Vector $\text{best\_costs}$.
\quad $\text{best}$ solution.

\textbf{Ensure:} If this solution has the best overall cost, or is predicted to, record the solution and costs and return \texttt{T}; else return \texttt{F}.

1: $\text{costs} \leftarrow []$
2: $\text{better} \leftarrow \texttt{F}$
3: \textbf{for} $i \in \text{length}(M)$ \textbf{do}
4: \hspace*{1em} \{Ascertain predicted cost for this factor.\}
5: \hspace*{1em} $\text{costs}[i] \leftarrow M[i](S)$
6: \hspace*{1em} \{Ascertain if this is better than the current best.\}
7: \hspace*{2em} \textbf{if} $\text{costs}[i] > \text{best\_costs}[i]$ \textbf{then}
8: \hspace*{3em} \textbf{return} \texttt{F}
9: \hspace*{2em} \textbf{else if} $\text{costs}[i] < \text{best\_costs}[i]$ \textbf{then}
10: \hspace*{3em} $\text{better} \leftarrow \texttt{T}$
11: \hspace*{2em} \textbf{end if}
12: \textbf{end for}
13: \{Solution must be better in at least one cost factor.\}
14: \textbf{if} $\text{better}$ \textbf{then}
15: \hspace*{1em} $\text{best\_costs} \leftarrow \text{costs}$
16: \hspace*{1em} \textbf{return} \texttt{T}
17: \textbf{else}
18: \hspace*{1em} \textbf{return} \texttt{F}
19: \textbf{end if}
CHAPTER 5. COLLABORATION AS SYNCHRONISATION

of non-dominated solutions, in which conflicts between the different metrics would be explored in order to offer a plurality of possible solutions. Computing the Pareto-optimal solutions is a desirable goal, though the search space is very large—and many directional cues can be drawn from users as to the desired constraints and goals, as discussed in the next section.

5.6 Pragmatic Solution

In practice, despite the various techniques for robustly decreasing the number of solutions that needed to be examined, the search space is still vast, as adding more users and T-devices increases the space exponentially. A solution that is less computationally-expensive is required, particularly as ad-hoc collaborations may occur frequently and increasingly on lower-power devices such as smartphones. In the following chapter, techniques for learning a great deal about a person’s capabilities and preferences (with respect to adaptations; the manner in which data is rendered) are developed. This knowledge should be used to minimise the search space.

In many other systems, such as SUPPLE and PUCs, one user interface is generated for one user on one device at a time (albeit, often on a different device or platform at different times). Given the size of the search space, a practicable technique was required in order to recognise the transient nature of collaborative situations and arrive at an acceptable solution in reasonable time (bearing in mind that, as an “always-on” system, possibly operating at least partly on embedded hardware, the reasoning process must be as efficient as possible). Two key principles were adopted.

React to user behaviour. Users will already have the best idea of which devices they wish to use for a given task. Reacting to this will produce a solution that feels more “natural” to the users involved as well as dramatically reducing search time. In order to find more “exotic” solutions, a full search could still be run in the background.

Seed the search. This can be done through both: (a) attempting to preserve the current set of entity–T-device assignments already in use whenever a new user and set of devices is added to a collaboration and (b) seeding the search with known-good solutions from a given user’s history and/or aggregated from a whole user group’s historical data.

To illustrate the potential to reduce the solution space, consider the two-user collaboration example above. The set of potential assignments of shared output
T-devices to each entity would be \( \mathcal{P}(\text{shared_output_T-devices}) \).

\[
\{(\text{}), (\text{Screen}), (\text{Speakers}), (\text{Screen, Speakers})\}
\]

If the fully-sighted user were already using the application, then the assignment of shared output T-devices selected may be \{ (\text{Screen}) \}. Should the blind user wish to collaborate on the same shared computer, then the four potential assignments can be reduced to just two (building upon the existing assignment, in order to keep the interface stable).

\[
\{(\text{Screen}), (\text{Screen, Speakers})\}
\]

This reduces the search space for this entity by a factor of two. Assuming this filtering can be applied to each entity, it will reduce the search space size given in Equation 5.2 by a factor of \(2^{\text{entities}}\). In general terms, and if the shared input T-devices are also similarly filtered, the size of the search space from Equation 5.2 can be reduced further.

\[
\text{space reduction factor} = \left(\frac{o}{o'} \times \frac{i}{i'}\right)^e
\]

where \(o' < o\) and

- \(o\) is the number of shared output T-devices.
- \(o'\) is the number of shared output T-devices after filtering.
- \(i\) is the number of shared input T-devices.
- \(i'\) is the number of shared input T-devices after filtering.
- \(e\) is the number of entities.

Continuing the “trivial” example above, \(1.18 \times 10^{21}\) may be reduced by a factor of \((2 \times 2)^7 = 16,384\) giving a search space of \(7.2 \times 10^{16}\).

Finally, this search space may be dramatically reduced by considering the users’ own selection of devices for the task. If the users wish to work together, then the collaboration would ideally take place on the shared computer system, rather than employing any of the users’ personal devices. The search can be carried out to with this goal in mind first in order to find a solution. This yields a considerably smaller search space.

\[
s = (2^i \times 2^o)^e
\]
where

\[ i \] is the number of shared input T-devices.
\[ o \] is the number of shared output T-devices.
\[ e \] is the number of entities.

In our running example, this reduces the search space to only 16,384 solutions to begin with.

Should this prove inadequate, a user may indicate a desire to use a personal device in combination with or instead of the shared computer, and the search could be seeded appropriately. This effectively allows the search to grow linearly on-demand as opposed to exponentially a priori.

### 5.7 Redundancy and Replacements

The most coarse forms of adaptation are the replacement (or removal) of content, or the duplication of that content across modalities (perhaps in order to reduce error rate [14]). These adaptations would almost certainly be made in advance of any interface being rendered, consummate with known high-level user abilities and preferences—for example “User 2” in the scenario used here. The user would not appreciate these as adaptations, as their application is pre-rendering, thus fitting in with the goal of adaptations remaining unobtrusive and not introducing apparent instability.

Replacement “adaptations” would be achieved as part of the search process described here, specifically by discounting any inaccessible T-devices from the pool of devices associated with a given user and by using an appropriate ordering function when searching the space of shared T-devices.

Redundancy could be achieved in two ways: either using an ordering function that explores the power set of T-device assignments “in reverse”—i.e. longer members of the power set first—and also by specifying constraints to force assignment of entities to more than one T-device, either shared or user-specific. Certain T-devices are designed for broadcasting (e.g. screens, loudspeakers) and others for personal use (e.g. headphones). Should other users be in the collaboration, perhaps whom do not wish for multimodal output, then such personal devices can be employed to satisfy each user’s constraints.
5.8 Summary

This chapter has cast the problem of collaboration between users of different capabilities (and with different devices) in terms of searching for a solution (set of assignments of users and devices to application parts) that fulfil the following conditions.

- That the people involved can perceive the entities of the application and changes in state.

- That the desired nature of the collaboration, defined in terms of a number of general factors, is supported.

So far, none of the reasoning processes developed is able to cater for minor-to-moderate impairments; they are mostly focussed on generalising a decision-making process surrounding current ATs, designed for people facing more severe and static accessibility barriers. However, with the addition of the concepts developed in the following chapter, the resolution of the processes will be improved to enable modelling of more general, less clear-cut situations.
The previous chapter developed a very basic reasoning process, based on interactions that were deemed possible. This type of reasoning process may be suitable for traditional ATs intended for people with recognised disabilities, but it is not necessarily of use to those facing minor-to-moderate impairments. This chapter extends and refines the reasoning process to consider the potential quality of information transfer during interactions, resulting in decision-making with much broader applicability.

Techniques for bridging between human capabilities and preferences and the nature of devices and available adaptations are introduced. These techniques are then used to: (a) reason about the effects adaptations appear to have, in given situations, for individual users (b) consider the interference between adaptations for a given user and (c) track changing capabilities over a period.

6.1 Overview

This chapter introduces reasoning regarding adaptations. An “adaptation” may be thought of in different ways, as follows.

End-users with minor-to-moderate impairments would not necessarily perceive an individual adaptation, as this may be a slight modification to a number of small parts of the system (such as an increase in font size or changes to other widget parameters). People with either more severe impairments, or whom are using legacy systems that are not as malleable may see adaptations as “plug-ins” to their system or traditional AT products that solve a particular problem.

Individuals would likely have their own favourite adaptations for a given impairment or situation, possibly expressed as a ranking (e.g. increased widget size over screen magnification; the replacement of the mouse with the
arrow keys for scrolling; alternative content substitutions or a change of rendering modality, such as from visual text rendering to TTS).

Developers may see adaptations as a range of different plug-ins or ATs that could be used to solve a given impairment (i.e. more than one could be used, and in any suitably large set of users, would be).

The reasoning process sees adaptations in two ways.

Concrete adaptations are the actual plug-ins and ATs as above and may be classified according to their effects on the system and mechanisms for achieving those effects.

Abstract adaptations represent the fundamental purpose of a given concrete adaptation in relation to information transfer, independent of its modality and mechanism. These will be introduced shortly.

The role of adaptations is very simple: to improve the transfer of information from the system to the user and vice-versa. In this light, there are two broad ways of thinking about adaptations.

Enabling adaptations are always necessary and enable a user to perform a certain task or perceive some certain data.

These adaptations are both mandatory and absolute, or “atomic”, so consideration of their effects on the communication between user and device need only be given at the most fundamental level. Examples include those seen in the previous chapter: translation or substitution of content in order to render it at all perceivable to the user.

Enhancement adaptations are not necessarily absolutely required, but have the potential to greatly improve a user’s experience.

As these adaptations more subtly change the way a user may interact with a system, reasoning about and assessment of them must be more sophisticated. Examples may include the use of TTS by someone who can, but prefers not to, read text from the screen and summarisation of content to enable it to be read more quickly.

As discussed in previous chapters, enhancement adaptations are key, due to their potential to improve accessibility in mainstream use and for people with minor-to-moderate impairments. Reasoning about adaptations, to this end, is carried out at different levels, as follows.

Individual users are affected by and interact with adaptations in two main ways.
Figure 6.1: Strands of adaptation reasoning. The two main areas (information transfer and feedback loop) are explored throughout the rest of this chapter.

**Feedback loop.** Tracking a user’s capabilities and preferences passively, by observing the adaptations the user instigates and monitoring feedback on adaptations, as well as capabilities directly where possible.

**Information transfer.** Using knowledge about a user to predict or suggest adaptations to the user based on problems detected, with the goal of affording the user the best possible access to the application and content.

**Collaborative** situations involve two or more people working together, in which case the adaptations they require need to be compatible. Hence there is a need to ensure that this is the case, or attempt to resolve the situation if incompatibilities are detected (if possible).

The two strands of interaction between an individual and the reasoning process are illustrated in Figure 6.1, in which the outer shaded area represents the feedback loop and the central shaded area the information transfer aspect. As modelling information transfer is simpler, it will be addressed first—although in practice both would be required to work in concert.
A third task for each individual—that of calibrating the system with some set of suitable starting values for, as well as periodic checks on, capability levels—must also be carried out. This is part of Sus-IT as opposed to the reasoning process, as was discussed in subsection 3.5.1.

Finally, it is important to note that, whilst some of the processes developed in this chapter rely on historical data, the perspective on all of the reasoning is from an instant in time. The feasibility of reasoning in a temporally-aware manner is explored in the next chapter.

6.1.1 Outline
As discussed, this chapter covers a number of themes relating to adaptations.

- Section 6.2 puts the reasoning techniques in context by describing how a real-world user would come into contact with the process.
- Sections 6.3 to section 6.7 introduce refinements to the existing theory to enable it to describe adaptations and their effects.
- Sections 6.8 to section 6.10 explain how the user capability and preference data, assumed to exist until this point, are actually discovered—including ascertaining preference differences with “context” or situation.
- Section 6.11 introduces the process for solving a given problem, at an instant in time, by finding and evaluating appropriate adaptation combinations.
- Sections 6.12 demonstrates the similarity between solving an adaptation conflict problem for one user and setting up a collaboration between multiple people.
- Section 6.13 develops a process for learning how users’ preferences vary across usage scenarios.

6.2 Real-world Reasoning
Whilst this work concentrates on the internal technical reasoning process, it is important to put this into the perspective of an imagined real system that uses the process, such as that being developed for Sus-IT.

6.2.1 Ethical Considerations
Many important factors surround the application of this work; these are discussed in more detail in subsection 9.1.4.
6.2.2 Adaptation Initiative

As has been discussed, most adaptations are affected before the UI is presented to the user, as they reflect users’ accessibility needs. However, spontaneous adaptations may also occur. The feedback reasoning process is independent of whether the initial abstract adaptation is instigated by the person using the system, or the CAAR process within the system—though this would require some sort of predictive reasoning of the type introduced in both sections 6.8 and chapter 7. There are actually three ways for adaptations to be invoked, below.

- The need for an adaptation is predicted (usually this will invoke the anticipated most-useful adaptation).

- The need for an adaptation is detected (via sensors, user behaviour monitoring as in [39] or similar).

There are many examples of “reactive” environmental adaptations, including portable computers and telephones that automatically adjust their screen brightness according to ambient light levels. These processes are hard-coded, however, whereas CAAR has the potential to react appropriately to a much wider (and less predictable) range of environmental events.

- The user requests help (which may be directed at a particular device or application entity), in which case reasoning must be carried out to ascertain the most likely useful adaptations that could be offered.

A more detailed description of the expected interaction workflow, with respect to user-initiated adaptations, from the end-user’s perspective is given in subsection 3.5.3.

System-initiated adaptations are, as discussed, significantly more controversial. However, there are areas in which system-initiated adaptations may be well-received and it is important to bear in mind that the purpose of the feedback loop is to effectively mirror the user’s own actions. There are several ways in which system-initiated adaptations may be informed, as will be discussed throughout this and the following chapter.

6.2.3 General Problem Detection

Detection of general user problems at run-time is out of the scope of this work, but is far from implausible. The classic techniques for determining whether users are having problems, adapted from the literature, include: monitoring for entities or T-devices with which users stop interacting after a history of frequent use;
monitoring for repeated sequences of actions from users, indicating that the system is not behaving as expected [39] and giving users the option of requesting help, optionally with a parameter: direction as to the problem area/entity.

### 6.3 Classifying Adaptations

Concrete adaptations have many important properties and could be classified in a number of different ways. Common classifications include their functional effects, often expressed in technical terms [64]. The list below gives the (orthogonal) properties pertinent to this work.

**Scope.** The area of effect, either: *mutations* to rendered individual abstract entities; *modifications* to entire T-devices or *adjustments* to application settings.

**Mechanism.** The basic process by which the adaptation affects a change. Examples include changing *parameters* (such as application settings, or the brightness of a display device) or making *replacements* (such as alternative content, or substituting one input device for a more accessible one).

Modality may be thought of as a pre-eminent property; all existing “concrete” adaptations are designed to work in one particular modality—screenreaders and alternative keyboards being two examples—and are often also tied to specific hardware devices and software platforms. It has been argued that adaptations should be as focussed and unobtrusively-applied as possible. Splitting larger ATs into much smaller adaptations is a goal of this work and Sus-IT, though the smaller the adaptation, the more infrastructure that is required of the supporting OS.

As CAAR takes a high-level view of adaptations, taking one of the allowed values for each of these properties provides enough data to classify any concrete adaptation. Appendix D details the various possible values for each property and gives examples of adaptations of each type.

**Mutations** applied to individual entities: colour; widget style (e.g. menu items → buttons); reduction in content; folding; content substitutions; size; font size; alternative content substitutions.

**Modifications** applied to T-devices—and therefore all entities being rendered or input via the T-device: screen brightness; loudspeaker volume; speech output speed; keyboard repeat rate; sticky keys; double-click speed; full-screen magnification; switching entirely from visual to auditory or haptic output.
**Adjustments** applied to applications; often settings/preferences within the particular program, affecting its behaviour: switch locale or measurement units (improves user understanding); view settings (would also result in effective coarse mutations to the UI, such as changing the function or layout of different panes in the application’s window); streaming video from the screen of one device to another.

Subsection D.1.1 discusses a fourth type of adaptation, purely to content, which is becoming increasingly popular, particularly due to the advent of semantic, web-based, data services. However, not being concerned with direct user interaction, it is out of the scope of this work.

### 6.4 Relation to Information Systems Theory

As discussed in subsection 3.2.1, any information-transmitting system is influenced by the processes modelled by Shannon. In fact the classic information system architecture is highly relevant to CAAR: when considering the paths from user to device and vice-versa separately, there is both an information source and receiver, as well as points where errors may be introduced. The addition, in this work, is that adaptations may be introduced in order to mitigate against “noise” and errors (as in the central shaded area of Figure 6.1).

Whilst Shannon’s work can be used to precisely model information systems and investigate limits on transmission performance, CAAR is a higher-level approach (and would often not have sufficient data to model the system to such high accuracy). However, when attempting to identify problems and track them over time, it is useful to understand which part of the system those problems may affect. Figure 6.2 gives an update of the classic Shannon model (Figure 3.4) to reflect the entities involved in the present reasoning problem (introduced in subsection 4.2.1).

Channels, in terms of the information-theoretic model, are analogous to the multiple output and input paths between users and individual T-devices. In order for a user to be able to interact with a channel effectively, the channel’s level of capability to convey information must meet or exceed that required (by the information being transmitted, as affected by the environment).

The CAAR process is inherently less precise, as it must accommodate individuals’ perceptions of and preferences regarding adaptations—in order to “home in” on the ideal adaptations for a given person.\(^1\) The process of learning from and reacting to users’ behaviour is part of the feedback loop described from section 6.8. Channels, in CAAR terms, have two key properties.

---

\(^1\)The case of at a given time is dealt with in the following chapter.
Capacity is the absolute limit on the information carried by the channel. For example: in a channel between a user and a video screen, the video screen will have a fixed maximum number of pixels. The person involved will also have a maximum capacity to resolve visual information presented via the screen.

If required, a T-device may be used at less than its full capacity. In this example, users with limited visual resolution may benefit from a lowering of the capacity (pixel resolution) of the display, because the nature of video screens is such that the pixels and therefore image would be enlarged.

However, there is a trade-off: lower-resolution displays may not have sufficient capacity to display all of the required information, resulting in documents that must be scrolled, or dialogue boxes that cannot fit onto the screen in their entirety.

Bandwidth is the rate at which information is transferred over the channel in a given unit of time. This would usually be the rate indicated by the capacity of the T-device, as above. However, it may be lowered independently of capacity, if required.

Continuing the display example: an adaptation that performs a full-screen magnification reduces the bandwidth of the display—only a fraction of the information being sent to the display is actually rendered at a given time. The capacity (resolution) of the display is not reduced; rather the bandwidth is limited in order to help the user perceive information at a more manageable rate. The trade-off here is that scrolling will certainly be required, though problems of interface elements or content not fitting on the screen would
likely be averted.

These properties are useful in predicting the relative utility of given T-devices for a particular user. They also indicate that, whilst CAAR aims to maximise the amount of information transferred, this actually means the amount of information perceivable by the user, giving rise to the following axiom of the reasoning process.

**Axiom 1** (Perceivable, over actual, bandwidth). *For a given channel, perceivable bandwidth is determined by the capabilities of user and T-device (it is the lowest common denominator). It vital to optimise a channel for perceivable bandwidth, as opposed to sending information at the maximum possible bandwidth of the user or T-device alone.*

### 6.4.1 Time

There are a number of different ways in which time may be considered, as follows.

**Rendering time** by a T-device is usually negligible. It generally only becomes significant when rendering data of more than $n$ dimensions on an $n$-dimensional device.

E.g. the time required to play a musical piece.

**Consumption time** is the time required to manoeuvre about the data in order to perceive it.

E.g. the *extra* time that having to scroll imposes on the task of reading some text. If the content text is enlarged, then more scrolling will be needed relative to another person’s rendering of the same content.

**Cognition time** is the time required to understand the information (largely out of scope of this work).

Continuing the example above: scrolling does also impose cognitive limitations. However the main component of cognitive time is that required by the brain to process the information into understandable thoughts.

**Reaction/response time** is that required to cognitively form a response to the information and actuate this (e.g. by using one’s limbs) and is equally out-of-scope.
6.5 Channels

A channel can be thought of as a one-way (output or input) connection between a particular person and a particular T-device. The channel may be thought of as having sub-channels (e.g. visual acuity and colour perception are both capabilities of the visual modality, so may be thought of as sibling sub-channels). Both the user and T-device have capabilities that may constrain channels; sometimes one will have noticeably greater capability than the other (e.g. loudspeakers can output sound with a larger frequency range than humans can detect; whilst the human eye can perceive far more colours than a typical display screen can produce).

The level of usable capability for a given channel or sub-channel (both referred to from now on as “channels”) is dictated by the lowest common denominator—be that the person or the T-device. Figure 6.3 provides an illustrative example. Note that as no extant device has cognitive capabilities, those available are simply those of the human involved (the requirements for cognitive capabilities would come from the content being presented). Also of note is that the capabilities are expressed in human terms, rather than device-based terms, as in subsection 3.2.3.

Sometimes it is possible to measure the level of capability in a given channel accurately. In fact this is almost always the case with devices, but far less often the case with humans. To measure human capabilities accurately often requires tests that would be considered invasive. However it is possible to detect when users are experiencing difficulties in certain channels; the reasoning behind this will be developed shortly.
6.5.1 Channels and Capabilities

The information bandwidth that a channel can support is, as above, determined by the capabilities of both user and T-device. The more capable a person is at using an input device, or perceiving the output generated from an output device, the higher the throughput of information. Channel capability may be thought of in two main ways.

**Direct.** The bandwidth of a channel increases with user and T-device capability.

**Skill-based.** Users who are in possession of certain skills can make considerably more effective use of a given device. A prime example being typing: looking at the keys and using one or two fingers will almost always yield a level of bandwidth far below that which a touch-typist could achieve.

Skills may be seen as “bundles” of capabilities, linked to certain body components. This makes for less verbose specification of capabilities on behalf of device manufacturers or classifiers and is therefore of use to Sus-IT. It can also be of use to the CAAR process in that it can help decide which person may be best suited to providing certain types of input.

The nature of some channels allows their bandwidth to be assessed directly (e.g. typing for text entry; double-click speeds), whereas the bandwidth of other channels is more subjective (e.g. perceived display clarity, which will vary from person to person). It can be helpful, but must not be vital, for the reasoning process to be aware of such direct measurements.

Having established the nature of channel capabilities, the notion of capability requirements arises. The content or interface being presented (or data being input) impose capability requirements on the user—forming the Information and Environment (IE) requirements. The level of channel capability must meet or exceed the level of capability required; a comparison is shown in Figure 6.4. In fact, the principal mechanism of adaptations, which are not able to arbitrarily increase the channel’s (i.e. user’s and T-device’s) level of capability, is to reduce the level of capability requirement for the channel. Considering the manner in which capability requirements are formed gives rise to three further axioms of the reasoning process.

**Axiom 2** (Channel capability requirements). *The notional “sum” of capability requirements for all channels (and therefore all entities) must be met by the user’s capability level in the relevant channel.*

**Axiom 3** (Adaptation requirement). *Adaptations must be applied when the level of capability required exceeds that available. These could be seen as enabling adaptations.*
Axiom 4 (Adaptation prerogative). Adaptations may be applied at any other time. These could be seen as enhancement adaptations.

It should be stressed that the difference or shortfall in a channel has to be assumed immeasurable by the reasoning process. Though the goal is to maximise this gap, its existence can only be inferred indirectly.

Finally, it is the task of an adaptation to mitigate against accessibility gaps, by either improving the user’s capability (e.g. using reading glasses—out of the scope of this work) or, as is the focus of this work, moving the burden caused by the capability gap onto a device or devices in the electronic system.

6.5.2 Capability Requirements as Bandwidth Limitations

In any given channel, the user is either receiving information from, or sending information to, a T-device. The capability requirements of the device, impairments on behalf of the user (which would lower the user’s capability level) and constraints imposed by the environment are considered as blocks to the unrestricted flow of information from device to user—sources of noise, or errors, in more information-theoretic terms.

An ideal environment would impose no requirements on the user, as there would be no barrier between user and the information that the device is rendering. Such an environment is unlikely to exist, however. An example of a close-to-ideal environment for one channel may include a darkened room for someone with light sensitivity. The analogy of “wind behind the sails” may appear to imply that some environments can have a positive effect (i.e. not only impose no requirements, but also actively help the user). However, this is a false analogy because here the focus is on information transfer, which could not be sped up—only impeded—by the environment, whereas in the sailing scenario, the focus is on carrying out a task
that involves interaction with the environment. The same is true of devices.

Adaptations, which would aim to decrease such barriers in one channel, may actually increase them in other channels. When using full-screen magnification to improve the user’s perception of visual output, a requirement for additional scrolling is often introduced. This will impact upon the user’s cognitive and motor capabilities by introducing extra requirements in those channels.

**Axiom 5** (Adaptation interference). *Adaptations in one channel may affect other channels (possibly, therefore, different modalities).*

### 6.5.3 Mitigating Capability Gaps with Adaptations

As above, the fundamental effect of an adaptation is to “move” a capability gap from one area to another (e.g. enlarging text to counteract visual acuity problems). Adaptations may also “spread” such a gap to reduce its effects (e.g. panning about an enlarged screen display, placing load onto the cognitive and a motor channel).

In terms of moving capability burdens between actors (users and T-devices), the following types of adaptation exist.

**User or Environment → T-device(s)**. Initiated either manually or automatically by the system in response to a detected or declared capability gap (e.g. use of larger buttons to improve mouse usage).

Environment-related adaptations would likely be initiated as a result of sensor input, devices may be adapted to improve their fit within the environment (e.g. adjusting screen brightness in reaction to ambient light; initiating an inverted-colour “night mode” on GPS navigation devices).

(Users may, of course, delegate tasks to each other, effectively moving potential accessibility burdens amongst themselves.)

**T-device → T-device(s)**. Moving the burden to a T-device may not be enough to solve the problem—the device or devices may be unable to support the entire burden itself (e.g. a zoomed-in screen must be scrolled, or may be unable to output at the required level of brightness). Further adaptations may be necessary to mitigate the effects of the burden on the device, potentially spreading it amongst other devices.

The familiar example of scrolling in response to a full-screen magnification or widget enlargement shows how the initial problem involving visual acuity is moved to a monitor, then spread to the motor channel, via a mouse perhaps. Should the user be unhappy with the increased motor load, a new set of adaptations would be needed to address this problem—this may involve
substituting the mouse for another device, or switching to rendering via audio instead of visually.

The key reasons for moving a burden into the T-device domain is that whilst the user or environment may be hard or impossible to adapt, the T-devices are under the control of the adaptivity system and, being electronic, are comparatively both easy and flexible to adapt.

Table 6.1 lists some key types of adaptation (to be formally introduced shortly) and categorises them according to the above, as well as their possible scopes (as in section 6.3) and whether they pass on capability burdens in the form of data (i.e. when data is moved to another T-device), or control of the system (e.g. scrolling).

## 6.6 Consequences and Abstract Adaptations

Adaptations aim to improve the conditions in a given channel, possibly at the expense of others. As will be discussed, there is a tension between improving one channel at a cost of negatively affecting the transfer of information from the system as a whole. A small number of system-wide indicators tracking information transfer are needed to provide a balance to this tendency, as follows. Table 6.2 gives the ways in which these indicators may change.

Global indicators are as follows.

**Information volume** provides an indicator of the amount of information that is being transferred between parties as opposed to the amount of information that could be transferred between parties. It is needed because each concrete channel seeks to maximise the amount of perceivable information transmitted; this may actually result in the use of adaptations (e.g. summarisation of content) that cause less information to be transmitted, albeit more clearly, to the person involved. For this reason, it is key to have a global indication of whether a solution is truly enabling the user to access as much of the data on offer as possible.

**Time** required to transfer the information must be taken into account at global level, for the same reasons. Whilst each concrete channel seeks to transfer the most perceivable data in the shortest time, the solution as a whole must have an indication of the time required to access the information. In a situation in which the user is pressed for time, it may be acceptable to reduce the amount of content rendered in order to speed up its transfer and processing.
Table 6.1: Coarse summary of possible adaptation usage and consequences: where might the need for adaptations arise; what are their possible scopes and how may they place burden on other T-devices in the system?

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>Barrier stems from</th>
<th>Scope</th>
<th>Burdens T-devices</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>U/E/D</td>
<td>T-device</td>
<td>O</td>
<td>µ</td>
</tr>
<tr>
<td>Aesthetic transform</td>
<td>•</td>
<td>o</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Scroll</td>
<td>o</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Fold</td>
<td>•</td>
<td>•</td>
<td>o</td>
<td>•</td>
</tr>
<tr>
<td>Reflow</td>
<td>•</td>
<td>o</td>
<td>o</td>
<td>•</td>
</tr>
<tr>
<td>Replace (Modality)</td>
<td>•</td>
<td>•</td>
<td>o</td>
<td>•</td>
</tr>
<tr>
<td>Replace (Abstract type)</td>
<td>•</td>
<td>o</td>
<td>o</td>
<td>•</td>
</tr>
<tr>
<td>Replace (Both)</td>
<td>•</td>
<td>•</td>
<td>o</td>
<td>•</td>
</tr>
<tr>
<td>Supplement (Modality)</td>
<td>•</td>
<td>o</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Bandwidth (channel total)</td>
<td>•</td>
<td>o</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Bandwidth (share of)</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
<tr>
<td>Reduce</td>
<td>•</td>
<td>•</td>
<td>o</td>
<td>•</td>
</tr>
<tr>
<td>Remove</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
</tr>
</tbody>
</table>

**Symbols:** • = yes; o = no. **Fields:** U/E/D = User/Environment/T-device; O = T-device modification; µ = abstract entity mutation.

**Note:** It is possible for an adaptation that does not directly cause a burden on another device to create a problem (such as lack of bandwidth) that ultimately requires another adaptation (e.g. scrolling) that does place burden on other devices.

**Note:** All “replace” adaptations may be carried out pre-rendering in order to present the interface in accordance with a person’s preferred learning style. These changes, which may be thought of as adaptations, would not be seen as such by the user.
Table 6.2: Possible consequences.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Indicator</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>Time</td>
<td>Increase</td>
</tr>
<tr>
<td>R</td>
<td>Redundancy (modalities)</td>
<td>Increase</td>
</tr>
<tr>
<td>I</td>
<td>Information (volume)</td>
<td>Increase</td>
</tr>
<tr>
<td>A</td>
<td>Aesthetics</td>
<td>Yes</td>
</tr>
<tr>
<td>L</td>
<td>Layout</td>
<td>Yes</td>
</tr>
<tr>
<td>S</td>
<td>(Bandwidth) Share</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Redundancy** across channels (i.e. repeating output, or accepting input, in different modalities simultaneously). Increasing input or output redundancy can reduce the rate of errors.

**Channel or entity** flags and quantities are as follows.

**Aesthetics** may be altered without altering layout; for example changing the colours of buttons to render them more readable to dyslexic people may assist them, and is unlikely to particularly unnerve a non-dyslexic person, as the interface appears largely the same (and motor memory may still be used).

**Layout** of the information presented. This is considered to be the spatial or temporal relationships between information (e.g. a toolbar being placed at the top of the screen). In collaborative situations, adaptations that affect layout for one user may present difficulty to another.\(^2\)

**Share, or bandwidth** of a channel, or the rendering capacity allocated to: applications; parts or entities rendered or input via that channel. The space or time allocated to render an entity from one part may be increased or decreased relative to the allocation for that part. In some systems, more than one application may be rendered on a T-device using techniques such as windowing, so the same may apply to applications as well as application parts. Finally, when applied to a channel as a whole, this modifies the bandwidth or capacity of the channel (thus, if information volume remains the same, affecting the global time-to-render indicator, too).

In many adaptation scenarios there is likely to be a compromise between one or more of these indicators—perhaps the most prominent is illustrated in Figure 6.5.

\(^2\)This is partly because some users may not perceive differences between toolbars and content, for example, so any adaptation affecting one but not the other may cause the application to appear radically different.
Figure 6.5: Time, Information volume and Space are orthogonal and must be traded against each other when extra capacity is required.

Whilst it is not possible to directly measure values for these indicators, it is possible to compare different solutions in terms of effects on them, as will be demonstrated. The orthogonality of these indicators is discussed in subsection 8.2.1.

### 6.6.1 Consequences of Adaptations

The notion of adaptations in one channel “interfering” with another is a key facet of the reasoning process; it is a phenomenon that can be used to help track changing user capabilities and indicate how the user perceives the effects of different adaptations. This section develops the modelling technique to allow links such as that above—between full-screen magnification and an increased burden on one’s cognitive and motor capabilities—to be expressed and detected.

Figure 6.6 depicts the concept of an adaptation being activated (by a person or the reasoning system). The initial adaptation is a zoom operation on a web page. This leads to a situation in which, due to the increase in size of the page, there is too little bandwidth to convey the same volume of data to the user at once. Either additional scrolling may be required, or a further adaptation may be carried out to reduce the volume of information on the web page (e.g. by summarising the content). The figure shows that in any given system there may be many possible concrete adaptations, but only the standard set of modalities and Time, Redundancy, Information, Aesthetics, Layout and Sharing (TRIALS) indicators described earlier.

**Axiom 6** (Adaptation consequences). *Adaptations have consequences; both positive and negative. These are likely to affect more than the channel in which the adaptation was applied. The application of an adaptation is therefore a trade-off between the positive consequences in one channel (and possibly others) and any negative consequences in others.*

**Axiom 7** (Mitigation of consequences). *Further adaptations may be applied to alleviate the negative consequences of an adaptation already applied.*
Note that in many legacy systems, the initial adaptation and a subsequent adaptation are inherently linked (e.g. full-screen magnification always results in scrolling over the output of the T-device). However, in more abstract systems, this is not necessarily the case; an increase in base widget size may be followed by any of a range of adaptations (e.g. resizing windows, scrolling in one or two dimensions, reading the text with TTS), perhaps even in combination.

6.6.2 Problems

The notion of a “problem” giving rise to one or more adaptations being invoked has been implied. A problem may have many causes, ranging from changing environmental conditions to user capability fluctuation, or applications in use. Fundamentally, problems are gaps, as in Figure 6.4, between what is required of a channel and what is available in that channel. It may be caused by either a lack of capability or a lack of capacity in a given channel (on the part of the user or T-device).

Examples include: lack of colour perception leading to lack of capacity in the vision channel (this is likely to only pose problems for some image entities); changing environmental conditions rendering it impossible to read a smartphone screen and lack of time on the user’s part, leading to too great a burden of information
Table 6.3: Sources of changes to channel capability and requirements.

<table>
<thead>
<tr>
<th>Source of change</th>
<th>Factor</th>
<th>Examples</th>
<th>When caught</th>
<th>Technical action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Known Channel</td>
<td>Person</td>
<td>Chronic conditions</td>
<td>Render-time</td>
<td>Apply adaptation</td>
</tr>
<tr>
<td></td>
<td>Available rendering space</td>
<td>Post-adaptation</td>
<td></td>
<td>Potentially insufficient bandwidth</td>
</tr>
<tr>
<td></td>
<td>Use/disuse of T-device</td>
<td>Run-time</td>
<td></td>
<td>Move affected entities; suggest re-rendering entire system</td>
</tr>
<tr>
<td>Unknown Section</td>
<td>Person</td>
<td>Some capability fluctuations</td>
<td>Adaptation, or help request</td>
<td>Request problem scope; suggest best-known adaptations</td>
</tr>
<tr>
<td></td>
<td>Personal adaptations (e.g. reading glasses)</td>
<td>Post-adaptation</td>
<td></td>
<td>(Effects: an apparent gain in capability.)</td>
</tr>
<tr>
<td>Known Info.</td>
<td>Application switch</td>
<td>Run-time</td>
<td>Render; apply chronic adaptations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mechanical test of rendering vs. user capability (e.g. known-good font size)</td>
<td>Post-adaptation</td>
<td>Automatically/suggest widening adaptation scope (e.g. update global minimum font size)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Some “stuck” situations</td>
<td>Run-time</td>
<td></td>
<td>Apply/suggest adaptations; link to (and adapt) application help</td>
</tr>
<tr>
<td>Unknown Info.</td>
<td>Gross physical change (e.g. light levels)</td>
<td>Run-time</td>
<td></td>
<td>React conservatively, or apply/suggest potentially more disruptive adaptations</td>
</tr>
<tr>
<td>Unknown Env.</td>
<td>Other “stuck” situations</td>
<td>Help requested</td>
<td></td>
<td>Apply/suggest adaptations; link to (and adapt) application help</td>
</tr>
<tr>
<td></td>
<td>Barriers where mechanical capability-matching tests are not possible</td>
<td>Adaptation, or help request</td>
<td></td>
<td>Request problem scope; suggest best-known adaptations</td>
</tr>
<tr>
<td></td>
<td>Situation urgency</td>
<td>Adaptation, or help request</td>
<td></td>
<td>Request problem scope; suggest best-known adaptations</td>
</tr>
<tr>
<td></td>
<td>Subtle physical change (e.g. light hue)</td>
<td>Adaptation, or help request</td>
<td></td>
<td>Request problem scope; suggest best-known adaptations</td>
</tr>
</tbody>
</table>

**Notes:** “known” in this case refers to entirely passive methods; “stuck” situations may be detected using Frohlich’s proposed technique [39]; sensor-related entities refer to commodity device capabilities.
presented by a long document. Table 6.3 summarises how changes to the capacity/requirements balance may be introduced and which are passively detectable. The examples given may apply to the whole channel or certain entities presented or input through it.

Extra capacity may be afforded or reclaimed (up to the maximum possible for a channel). This may occur when: environmental conditions become more favourable; extra T-devices become available or when users perform personal adaptations external to the reasoning process.

### 6.6.3 Introducing Abstract Adaptations

All adaptations fundamentally work by affecting the transfer of information in some manner. When the “too little bandwidth” consequence, as above, occurs, the capability requirements of the information being transmitted can be reduced in four different ways. Each of these can be considered as an abstract adaptation and would be afforded by any suitable concrete adaptation—*in any modality*. The possible abstract adaptations (along with their counterparts in a more formal information theoretic modelled system) are as follows.

**Channel, part or entity** adaptations

**Aesthetic transform.** Change the manner in which the information is presented, without changing the volume of information or its abstract type. Such adaptations are designed to ease cognitive processing and do not impose a burden on any other channel. Examples include: modifying screen brightness or speaker volume and “night mode” on GPS devices, which presents map data in inverted or subdued colours, so as to avoid distracting the user. Input “aesthetic” adaptations may include increasing or decreasing sensitivity of a device so as to lower the cognitive and motor burden on the user.

*Analogous to altering the encoding of information in the channel in a more acceptable way for the user.*

**Layout transform.** As with aesthetic transforms, but altering the spatial or temporal internal structure of the information.

*Analogous to altering the encoding of information in the channel in a more acceptable way for the user.*

**Scroll.** Provide means for the user to navigate about the original content on a lower-bandwidth channel (or area thereof). In channels with more than one dimension, scrolling could be in any number of the dimensions.
People may find one-dimensional scrolling much more acceptable than two-dimensional scrolling, due to the lower cognitive and motor load this entails.

*Analogous to taking more time to compensate for a lack of capacity in a channel.*

**Fold** or “collapse” parts of the original content in order to present more of an overview to the user; allow the user to expand upon parts that are interesting. This may be done in two (popular) ways, as follows.

**Fold/Unfold.** Reduces content initially, then, at the user’s request, selectively re-expands it. Scrolling is still needed (1–n-dimensional).

**Fold-in-place.** Makes a whole series of alternative application parts share the same rendering time/space. Scrolling is not explicitly needed (but may be depending on the size of the rendering space that could be allocated).

*Analogous to taking more time to compensate for a lack of capacity in a channel.*

**Reduce.** Remove content (e.g. by summarisation) or interface elements deemed to be non-vital. This will almost certainly require some cooperation on behalf of the interface designer or content author, as automatic summarisation may not yield adequate results. (After such an operation has taken place, the possibility for re-expansion exists.) Channels may be removed from the system if a user is no longer able or willing to make use of them (though if this is caused by a chronic disability, such removal would have already been taken into account in the rendering process.)

*Analogous to sending less information.*

**Expand.** (Reverse of Reduce—may be carried out if the user has more time, or the channel has more capacity, available.)

*Analogous to sending more information.*

**Share** of rendering space or time devoted to this application, part or entity may be increased (or decreased)—e.g. resizing a window of one application to accommodate an increased content font size (at the expense of rendering space or time devoted to other applications in the same channel).

When applied to a channel, this adjusts the bandwidth of the channel, potentially altering the level capability required to perceive it.

*Analogous to changing the capacity of the channel.*
**Replace** an entity for an alternative either in one of three ways, as follows (as discussed in section 5.7, this would normally take place before rendering the interface, dictated by user preference).

- In the same channel (therefore modality), but with a different abstract type (e.g., replacing an image with text, but still rendering the text visually).
  
  *Analogous to changing the way the information is encoded.*

- In a different modality, keeping the same abstract type (e.g., changing visually-rendered text into aurally-rendered text).
  
  *Analogous to sending the information down a different channel.*

- In a different modality, with a different abstract type (e.g., swapping an image for its textual description and rendering this using TTS).
  
  *Analogous to sending the information down a different channel.*

**Supplement** the material in one channel with the same material rendered in another; rendering the information both on-screen and via TTS, for example.

*Analogous to sending data over multiple channels, thereby increasing redundancy.*

Figure 6.7 demonstrates scroll, fold and reduce being applied to both content, as in the example above, and an interface. Clearly users may have differing views as to which adaptations may be acceptable in different circumstances. Whilst most people would arguably prefer for reduce *not* to be applied to an interface, some may prefer the lower cognitive load that would result; it is the task of CAAR to accommodate, as opposed to stipulate, user preferences.

### 6.6.4 Possible Abstract Consequences

As touched upon above, any concrete adaptation is in fact an embodiment of an abstract adaptation. This means that the concrete “web page zoom” given as the starting point in this example was, itself, an embodiment of an abstract adaptation (bandwidth share; applied to the web page content only).

Any instance of one of the above abstract adaptations will have effects in three areas, determined both by the nature and scope of the adaptation as well as the application, users and T-devices.

- Intended positive effects in the source channel. Either reducing the IE requirements, or reducing the bandwidth of the channel so as to render it more perceivable to the user.
Figure 6.7: Some possible actions when “too little bandwidth” results from an adaptation: for content (left) and UI (right). Users may prefer different abstract adaptations dependent on whether content or UI is being changed.
Reducing channel capacity/bandwidth may render a channel more perceivable but also render the current IE requirements too great for the channel—introducing a further accessibility gap, necessitating further adaptation.

- Effects of this change in terms of the TRIALS indicators (as in the following subsection).

- Burdens inherently placed on other channels. Depending on the circumstances, these burdens may constitute further problems in their own right. Purely aesthetic adaptations do not raise further problems though, as with all adaptations, the user can provide feedback on their perceived efficacy.

Note that each abstract adaptation is considered separately—scrolling, for example, is not an inherent part of zooming (as there are alternative adaptations to scrolling). However, most forms of zooming will introduce the further problem of “too little bandwidth” at either the part/entity level (such as changing the font size used for a web page) or over an entire channel (as with full-screen magnification).

**Axiom 8** (Disruption). *Adaptations have a range of possible consequences. When exploring potential adaptations for use in a given situation, it is preferable to evaluate the least disruptive first (those which have the least effects on other modalities and TRIALS indicators).*

A common pattern is that adaptations are invoked in pairs: the first addresses a problem encountered in one channel and may introduce a lack of bandwidth in that channel; the second places additional burden on other, more capable, channels to address the lack of bandwidth in the source channel.

**6.6.5 Indicator Consequences Example**

Figure 6.8 gives some different courses of action that could be taken when reading difficulty is encountered. Each can be modelled in terms of their effects on channels and requirements.

Two example initial adaptations and possible secondary adaptations are given in Table 6.4. The upper portion of the table summarises the situation in which full-screen magnification has been invoked. At this point, most systems would have (2D) scrolling around the now-virtual screen as a hard-coded response. However this is considered just one of a range of possible responses to the initial adaptation’s consequence. If the application in question is non-adaptable (i.e. legacy), then options are limited; only replacement of the entire application with an audio-based channel is an option.
Table 6.4: Example ranking of abstract adaptations for legacy and abstract scenarios (in order of preference of an individual person).

<table>
<thead>
<tr>
<th>A</th>
<th>Adaptation</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
<th>A</th>
<th>V</th>
<th>M</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td>i</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td>i</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td>i</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td>i</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td>i</td>
<td>e</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td>i</td>
<td>e</td>
</tr>
</tbody>
</table>

- Screen zoom (device)
  - TRIALS:  ○ ○ ○ ↓
- Scroll (2D)  ↑ ○ ○ ○ ↑ ↑
  ○ Supplement (A)  ↑ ↑ ○ ○ ○ ↑
  ○ Replace (A)  ↑ ○ ● ○ ↑ ↓
- Font size (entity)
  ○ TRIALS:  ○ ○ ○ ↓
- Resize*  ↑ ○ ● ● ↑
  ○ Scroll (1D)  ↑ ○ ● ○ ↑ ↑
  ○ Fold  ↑ ○ ● ○ ↑ ↑
  ○ Supplement (A)  ↑ ↑ ○ ○ ○ ↑
  ○ Scroll (2D)  ↑ ○ ○ ○ ↑ ↑
  ○ Replace (A)  ↑ ○ ● ○ ↑ ↓
  ○ Reduce  ↓ ↓ ○ ● ○ ↓

**Fields:** A = requires an adaptive application or content-manipulation API; TRIALS = indicators; A/V/M/C = channels (each with c, i, e levels). **Symbols:** ○/● = affects/does not affect indicator; ↑/T-device = increase/decrease.

*resizing the window may not always be possible (e.g. some platforms have only full-screen applications). Effectively increases channel capacity by increasing the share of the channel used to render the given application/entity.
If, however, the application is adaptable (i.e. has a mutable UI of the type defined in section 4.4), then alternatives to the original full-screen zoom adaptation exist; the system could replace the original full-screen zoom with an application- or OS-wide font-size increase.³ This would still result in there being too little bandwidth to display the required information, but, as shown in Table 6.4, more options for correcting the situation exist.

Individuals will have an order of preference with regard to these options (partly based on the severity of the adaptation and, therefore, its collateral effects—to be addressed shortly). It is necessary to rank the available adaptations according to a person’s preferences and the situation in which they may be applied. The feedback loop is intended to learn user preferences in abstract terms, such as the following.

- “1D, as opposed to 2D, scrolling is more important than preserving the layout of content.”
- “Retaining, as opposed to reducing, information is more important than avoiding 2D scrolling.”
- “1D, over 2D, scrolling is preferable (where supported by the application).”

³Many contemporary OSs partially support this mutation, but only for UI elements; not content too, as a true abstract system would.
CHAPTER 6. ADAPTATIONS AND CAPABILITIES

The preferences are used as in ranking for the various alternative abstract adaptations that could be employed in a given situation; there would likely be differences between content- and interface-related preferences (most users would likely prefer that “reduce” never be applied to UIs). Naturally the highest-ranked adaptations are preferred, but in the case of unresolvable conflicts (introduced later), lower-ranked options may have to be used instead.

Figure 6.9 illustrates how abstract adaptations have an effect on other channels by continuing the above example, showing the links between the different possible abstract adaptations, which determine where the adaptations’ burden is placed.

6.7 Real-world Adaptations

Note that multiple adaptations from Figure 6.8 could be invoked simultaneously, to varying degrees (e.g. variable font size as well as a certain level of full-screen zoom). It has so far been implicitly assumed that adaptations are discrete; e.g. that there is only one level of full-screen zooming. Also, the notion of multiple adaptations being invoked together to solve a problem has not been supported. Naturally, this does not fit the real world; CAAR must be able to take the following into account.

- The extent to which adaptations are—or, rather, could be—made.
- The extent to which different alternatives change the state.
- How multiple abstract adaptations may be of use.
- When should one “give up” on a purportedly higher-bandwidth channel and switch to a lower-bandwidth (possibly linear/non-random-access) channel that requires fewer adaptations?

To address these points, the scope of application of adaptations must be understood. Further the effects of abstract adaptations must be quantifiable, which allows them to be compared; i.e. it must be possible to ascertain the relative utility of a range of proposed adaptations with respect to a particular problem (consequence encountered in a given usage scenario, therefore encompassing the application, users and T-devices in use). This is done by considering abstract consequences as well as burdens placed on other channels.

6.7.1 Adaptation Scope and Indicator Consequences

The three main types of adaptation, by scope of application, are listed below, along with examples of concrete adaptations at each level of interference given
Figure 6.9: Linking capability problems to TRIALS and abstract adaptations. The technique involves exploring a tree and is entirely mechanical, so can be carried out automatically. It is also important to note that, as adaptations may introduce further problems, there is a risk of cycles occurring whereby a previously-solved problem is re-introduced. To prevent this, constraints are required to ensure termination (e.g. allowing abstract adaptations to be employed only once on a given path from the root to a proposed solution).
above. Any proposed set of adaptations for a given problem should aim to reduce negative effects on other channels, as well as changes to overall indicators.

**Mutations** (adaptations applied to individual entities)

**Requirements only:** colour; widget style (e.g. menu items → buttons)

**Requirements and indicators:** reduction in content; folding; content substitutions.

**Channel:** size; font size; alternative content substitutions.

**Modifications** (adaptations applied to T-devices—and therefore all entities being rendered or input via the T-device)

**Requirements only:** screen brightness; loudspeaker volume.

**Requirements and indicators:** speech output speed; keyboard repeat rate; sticky keys; double-click speed.

**Channel:** full-screen magnification; switching entirely from visual to auditory or haptic output.

**Adjustments** (adaptations applied to applications; often settings/preferences within the particular program, affecting its behaviour)

**Requirements only:** switch locale or measurement units (improves user understanding).

**Requirements and indicators:** view settings (would also result in effective coarse mutations to the UI, such as changing the function or layout of different panes in the application’s window).

**Channel:** streaming video from the screen of one device to another.

Table 6.5 gives a worked example of how a set of adaptations might change the state of the system, in terms of the TRIALS indicators.

### 6.7.2 Extent and Compromise

The CAAR process must be aware that the extent of adaptations can be altered. However, to have in-depth knowledge of the ways in which this may be altered, particularly for adaptations involving intricate models, such as colour deficit or motor disorder correction, would be duplicating work and unnecessary (as discussed in section 2.3).

Given that CAAR is concerned with combinations of adaptations being invoked, if a set of adaptations is invoked this is either because: the primary adaptation caused side-effects that the secondary one fixes; the primary and secondary
Table 6.5: Worked Example of adaptation application. The previous state is changed by the two applied adaptations according to their indicators.

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
<th>Type</th>
<th>Modality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>atype(e) good</td>
<td>modality(e) good</td>
</tr>
<tr>
<td>B</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>o</td>
<td>o</td>
<td>↑</td>
<td>atype(e) good</td>
<td>modality(e) good</td>
</tr>
<tr>
<td>B</td>
<td>↑</td>
<td>–</td>
<td>–</td>
<td>o</td>
<td>•</td>
<td>–</td>
<td>atype(e) good</td>
<td>modality(e) good</td>
</tr>
<tr>
<td>Together</td>
<td>↑</td>
<td>–</td>
<td>–</td>
<td>o</td>
<td>•</td>
<td>↑</td>
<td>atype(e) good</td>
<td>modality(e) good</td>
</tr>
</tbody>
</table>

**Symbols:** ● = yes; ○ = no; – = not applicable.

**Note:** the fact that both adaptations were applied together implies that: (1) the situation afterwards is an improvement on the situation before and (2) that the situation afterwards is an improvement on both situations that would have followed had only one adaptation been invoked.

Table 6.6: Compromise in real-world adaptations. Two adaptations are employed simultaneously, to different magnitudes, in order to mitigate the negative effects of each.

<table>
<thead>
<tr>
<th>Zoom</th>
<th>Font size</th>
<th>Positive effects</th>
<th>Negative effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>None</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>High</td>
<td>None</td>
<td>Content layout preserved.</td>
<td>Lots of scrolling (panning).</td>
</tr>
<tr>
<td>None</td>
<td>High</td>
<td>Only one-dimensional scrolling (within content) required.</td>
<td>Content layout affected.</td>
</tr>
<tr>
<td>Medium</td>
<td>Medium</td>
<td>Larger viewport (and less panning) than high zoom alone; legacy interface adapted somewhat.</td>
<td>Legacy interface elements not adapted.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Two types of scrolling necessary (panning and content).</td>
<td></td>
</tr>
</tbody>
</table>

Adaptations together, possibly to a lower extent than alone, afford the same level of accessibility but with less burden on the user or for arbitrary reasons the user prefers this combination of adaptations.

The reasoner must be aware that the extent of adaptations can be changed—in fact, adaptations inherently report back to the reasoner when this occurs, so that it can maintain the capability range/change mappings. However it is not necessary for the reasoner itself to apprehend detailed information about the available extents of adaptations. Figure 6.10 gives an example of how two adaptations together can yield a less disruptive but equally accessible outcome—for a specific user—and this is discussed more in Table 6.6.

4Appendix B provides an overview of a reference run-time system.
Figure 6.10: Example of compromise in adaptations. Top left: none; top right: full-screen magnification; bottom left: font size increase; bottom right: both.
6.7.3 Alternatives

There will be a point at which it is no longer viable for a user to continue using a given channel due to heavy adaptation requirements. This point is not one that could be easily predicted. What is known is that, once an interface has been rendered for a user, it has been rendered according to that user’s preferences, so it is likely that a situation that has such dramatic effects to render the channel unworkable has been caused by unexpected environmental or situational changes.

Due to either embedded sensors in the device that can detect some such circumstances, and the interaction workflow proposed in subsection 3.5.3, the user should be able to communicate the problem to the system, or it be automatically detected. At this point, either “reflex” adaptations to counteract environmental problems or the process described in section 6.11 would be engaged.

6.7.4 Times at which Decision-making is Required

There are two distinct times at which adaptations may be required, as follows.

**Rendering time** involves adaptations being applied for the user before, or as, an interface is being constructed. These could be highly reliably informed by the user’s capabilities (particularly those related to any chronic impairments) and preferences, in terms of: presentation style; preferred modalities for given types of content; ability to use certain T-devices and TRIALS preferences.

**Run-time** may involve the detection or declaration of capability fluctuations or other problems, thus resulting in the need to choose sets of adaptations that are most suited to the new conditions.

Run-time problems may be detected or declared, based on input from different sources, as follows.

**The user** may inform the system of a problem directly. Direction as to exactly the areas of the system affected by the problem (e.g. “all text entities”) may be provided. The users are often the only actors who can reliably inform the system if an adaptation should be considered successful.

**The system** may be able to detect problems, such as: changing environmental conditions (via sensor); changes in application in use, or whether adaptations made to assist the user in one area of the system may reasonably be applied to another.
For example: would be possible to detect that a minimum font size, set in a web browser, exceeded the global minimum font size in the OS. Based on the user’s attitude to initiative, the system may then apply, or offer, a global minimum font size adaptation to the user (as in IBM’s WAT [91]).

6.7.5 Bootstrapping User Data

Throughout this section, there has been the implicit notion that the user’s capabilities and preferences are available for the decision-making process to use. This is clearly not always the case, in theory, because a person has to begin using the system at some point, before such a history of data can be collected (even if it is built into the OS).

In practice, any production system would aggregate anonymised user data to improve many aspects of the system’s operation (such as suggestions for adaptations in given scenarios, based on other users’ experiences). Some initial basic calibration, based on either direct questions regarding users’ abilities, presenting them with some of the “mini-games” (Figure 3.16) and analysing the settings in their system, would provide sufficient information to match the new user to a “default” profile. Technical processes to achieve this may include inference rules, or taking aggregate data from already-active users. Other work-packages in SusIT are developing complementary approaches to eliciting users’ learning support needs.

6.8 Monitoring Capabilities and Preferences

This is the feedback loop portion of the reasoning process. It is slightly more complex than as introduced in Figure 6.1, to encompass channels and the IE capability requirements. The actual feedback loop is shown in Figure 6.11 (in which the central information-transfer reasoning from Figure 6.1 would take place inside the CAAR block).

The notion of relative capability in a channel, versus the capability requirements of that channel, has now been made explicit. Whilst it is acknowledged that it is not actually possible to measure the difference between the two (as this is down to individual users’ perceptions), ensuring the requirements do not exceed user capabilities is the central task of CAAR—thus introducing a further layer of indirection to the control of the process. Fortunately, however, it is possible\(^5\) to monitor users’ reactions to the changes made by adaptations, thus providing needed data on adaptation side-effects and preferred T-devices.

\(^5\)at least when working with people with minor-to-moderate impairments
Figure 6.11: The feedback loop part of the reasoning process. Elements “below” the dotted line are asynchronous (controlled by the user or environment).
Three principal means of bringing data into the system are used, as below. A user’s profile would be built up over time by logging events of the following types. The rest of this work is concerned with the latter two sources of data.

**(Active) calibration.** As discussed in subsection 3.5.1, Sus-IT is developing a range of “engaging” versions of classical capability tests. These are used to engage with new users and roughly gauge capability levels without putting undue pressure on the user. Techniques could be used to create a standard user profile for a new user, based on aggregation of anonymised data collected so far on other users.

*Feedback informs: capabilities.*

**Passive monitoring.** As discussed, it may sometimes be possible to monitor the bandwidth of information transfer being achieved through certain T-devices (often input devices) such as keyboards and mice (in fact, the double click speed may even be tracked over time).

*Feedback informs: capabilities.*

Additionally, the system in development allows users to request assistance at any time. The request for assistance may be directed to specific areas of an application (the more abstract the UI, the more specific this feedback can be). This can be of use in determining which T-devices, data types or applications a user is finding troublesome.

*Feedback informs: the adaptation base; capabilities.*

Finally, the manual invocation of an adaptation by the user implies that the adaptation was deemed necessary at that time and—assuming it is not rejected shortly afterwards—implies particular capabilities that were causing problems.

*Feedback informs: the adaptation base; capabilities; usability/accessibility metric.*

**Adaptation feedback** is collected via an always-available UI. Adaptations that have already been applied may have accept/reject feedback given. When the user invokes a given adaptation (which, for adaptations internal to the system, could be detected via passive monitoring above), this provides data not only on the adaptation invoked but on any alternatives not invoked. Further, the system is able to offer adaptation choices to the user, if desired. Likewise, the choices of adaptations selected (and those discounted) are of use.

*Feedback informs: the adaptation base; user preferences.*
Figure 6.12: The cycle of adaptations and capabilities informing each other. Either an invoked adaptation, or a capability measurement may trigger the rest of the cycle at any time. Depending on user preferences with respect to adaptation initiative, suggested adaptations may be automatically invoked, or held until the user requests help.

**Usability/Accessibility hitrate data** is implied when the user requests help with, or invokes an adaptation upon, a particular application part.

*Feedback informs: the metric from section 4.1.*

The cycle of adaptations informing user capabilities which, in turn, inform adaptations is depicted in Figure 6.12. Over time, data on adaptations invoked (and rejected) and capabilities would build up into a profile. Aggregation of anonymised data from other users would be employed in a production system, to inform possible adaptations for a user with a nascent profile. The monitoring process is responsible for tracking three similar areas, as follows.

**Capability data** from adaptations invoked—and using this to suggest other adaptations (in line with user preferences, as below).

**Abstract adaptation preferences** from adaptations invoked—and using this to allow the user to discover alternatives based on those preferences (or using them in collaborative situations to satisfy more than one person).

**Context.** Users’ preferences will undoubtedly differ across usage scenarios (devices; modalities; applications and so on). Methods are needed to passively ascertain these differences.
Figure 6.13: Capability tracking: adaptations have capability implications. Over time, as adaptations are used, a picture of most-implicated capabilities emerges.

**Initiative.** Ascertaining if the system might reasonably take initiative to apply certain adaptations (based on either chronic impairments, detected environmental changes or habitual user behaviour).

### 6.9 Mapping Adaptations to Capabilities

The central premise of tracking adaptations is depicted in Figure 6.13. In the figure, an adaptation is instigated in response to a particular problem. This adaptation aims to address a certain set of capability problems (in turn, it may have effects on other capabilities and devices, as discussed elsewhere). At some later point, a further adaptation is invoked. It too aims to assist with certain capabilities. Whilst it may not be possible to directly measure user capabilities—and thus it is not possible to know which of the possible capabilities are in fact causing problems for the user—it is possible to build up a picture of likely-important capabilities over time. This approach is effectively a much-simplified version of the “Monte Carlo” methods, with the random generation of inputs is replaced with taking actual input from users over time (see section 8.4). There are several challenges, as follows.

**Cardinality.** Some adaptations naturally map to one capability, whereas others could affect many.

This problem is addressed by building up the picture of a person’s capabilities over time, thus involving many adaptation invocations.

**Specificity.** Some adaptations belie very specific capability problems (e.g. colour perception deficit), whereas others are more general (e.g. replacing one pointing device with another).
This problem is addressed by adopting a capability structure such as the ICF, which is tree-based (i.e. more general capabilities at the roots; very specific capabilities at the leaves). An adaptation could be seen to apply to “hearing” or “colour perception” for example.

Further, many adaptations will simply flag certain capabilities as being possibly related, whereas others (as with active calibration) may provide specific values (e.g. a visual acuity of 6/60).

Finally, some adaptations will provide concrete values, but in terms of a given device, such as preferred double-click rate. This belies some sort of difference in motor capability, and a value that can be used on a particular device or OS, but which is not portable across devices.

**Environmental noise.** The application of an adaptation may be due to environmental factors, as opposed to capability impairments on behalf of the user.

The potential problem of a fluctuating environment causing noise is acknowledged. It is possible normalising or damping the incoming data would avoid it being skewed by temporary phenomena; however if adaptations are brought about by environmental changes, these could well have been detected—and adaptations instigated—by the system.

The principal data structure involved is that of the capability classification (Sus-IT uses a structure similar to that of the ICF; see Appendix C). Any device-specific values, such as double-click speed or display brightness, may be stored or, better, dynamically generated and placed in an area addressed by a (user, T-device) tuple, so that they may be applied when the user logs in (as in various personalisation projects including SNAPI [102]).

Passive gleaning of capabilities is achieved very simply, by monitoring the use of adaptations. Adaptations may be invoked by the system or user, but as long as they are accepted (i.e. not rejected by the user), they contribute to the picture of users’ capabilities. As some adaptations cater for chronic conditions, whereas others are used more transiently, *engagement time* is a more appropriate counting metric than the number of invocations.

The developers of adaptations are called upon to indicate the capabilities relevant to an adaptation; this is considered reasonable, as it places little extra burden upon them. The result is a mapping between adaptations and capabilities, such as that in Figure 6.14. As in Appendix C, a standard capability classification, such as the ICF, is used (though this has no computational relevance). Table 6.7 gives the mapping in each direction. The graph would ideally be maintained on a
Figure 6.14: Example mapping from adaptations (left) to capabilities (right) (to be extended in chapter 8).

Table 6.7: Example adaptations and capabilities mappings (static across users).

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>Capabilities</th>
<th>Capability</th>
<th>Adaptations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
<td>2</td>
<td>A, B</td>
</tr>
<tr>
<td>C</td>
<td>3, 4, 5</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>4, 5</td>
<td>4</td>
<td>C, D</td>
</tr>
<tr>
<td>E</td>
<td>5, 6, 7</td>
<td>5</td>
<td>C, D, E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>E</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>E</td>
</tr>
</tbody>
</table>

central server, so will include adaptations that may be unavailable or unsupported on any given T-device. Benefits from holding such data centrally include discovery of uninstalled but support-able adaptations, as well as bootstrapping new devices, such as public terminals.

Adaptation usage data may be supplied to the adaptation-capability mapping in order to build up a picture of pertinent capabilities in a given situation, as depicted in Figure 6.13. Capability data may be more directly (and invasively) obtained by the capability-testing “mini-games” (Figure 3.16). Being an active method, this is more disruptive though can offer more accuracy than passive methods.

As has been discussed extensively, the capabilities gleaned by the system are not assumed to be users’ actual capabilities; rather they are the capabilities pertinent to the current situation. If users prefer to use reading glasses than request adaptations in order to read more clearly, then this would clearly be (a) not detected and (b) perfectly acceptable. The social challenge, undertaken partly by
Table 6.8: Example capability implications, based on adaptation usage, during a period of usage (user- and period-specific).

<table>
<thead>
<tr>
<th>Ranked capabilities</th>
<th>Usage amount</th>
<th>Normalised weighting</th>
<th>Log. weighting (unused)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>500</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.9: Scores for adaptations, given normalised weightings.

<table>
<thead>
<tr>
<th>Adaptation</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>$6 + 10 = 16$</td>
</tr>
<tr>
<td>C</td>
<td>$1 + 2 + 6 = 9$</td>
</tr>
<tr>
<td>D</td>
<td>$2 + 6 = 8$</td>
</tr>
</tbody>
</table>

Sus-IT, is to develop encouraging ways for users to access support, such as the “help button” (see subsection 3.5.3).

### 6.9.1 Capability-based Adaptation Suggestions

The immediate goal is to suggest adaptations that are available on the relevant T-device and are consummate with users’ preferences in order to assist with the capabilities gleaned above. The following facets are considered.

**Coverage.** One adaptation may support many capabilities. The fewest adaptations covering implicated capabilities should be used.

**Priority.** The relative importance of capabilities, based on supporting adaptation usage count, should also be reflected.

The algorithm for suggesting adaptations is given in algorithm 6.1.

**Algorithm 6.1** Adaptation suggestions based on ranked capability coverage.

- Maintain a ranked list of capabilities “touched” by the adaptations in use over a period (using an established mapping such as that in Figure 6.14).
- Score each adaptation, by adding together the weight of each capability it supports (Table 6.9 lists scores for each pertinent adaptation, based upon this).
Due to the use of normalised weights, it is possible for an adaptation that
does not support the top-weighted capability, but has wide coverage, to have a
higher score than an adaptation that does support the top-weighted capability. A
logarithmic scale for the weightings could be used to prevent this, however such
skewing of the weightings would be unrealistic, particularly given the assumption
of potentially multiple minor-to-moderate impairments—in such a case, there is
unlikely to be a clear “most-important” capability. Under normalised weights,
the resulting list of suggestions will be ordered sufficiently accurately—i.e. with
“good” adaptations clustered towards the start.

6.9.2 Mapping Capabilities to Adaptations

It may be appropriate to invoke certain adaptations instantly when a particu-
lar capability problem is detected—either due to environmental changes, such as
ambient light/sound levels, or to user conditions that are chronic, such as colour
perception deficit. This already occurs on most mobile devices. Such “reflex” in-
vocation of an adaptation would give the reasoning process information about the
environment’s potential effect on user capabilities and may, for example, result in
suggested or automatically-invoked aesthetic adaptations to applications (such as
“night mode” for GPS navigation embedded software as above).

6.10 Underlying Adaptation Consequence
Preferences

This section introduces techniques to reflect users’ preferences in terms of ab-
stract adaptations—thus the underlying consequences of given concrete adapta-
tions. There are three fundamentally important factors to note.

- Users may be unaware of, or unable to obtain, more helpful adaptations than
  those which are available, so any manually-initiated adaptation cannot be
taken as “perfect play” by the user.\(^6\)

- Opportunities for users to make genuine decisions are rare, because the
  primary goal of the CAAR process is to be passive. As a result, adaptation
  usage is the only solid data available for use in determining user preferences.

- In order to counteract this limitation, it is necessary to present users, when
  possible, with alternatives that will allow the broadest range of consequences

\(^6\)I.e. invoking one adaptation does not imply that it has been chosen above all other available
adaptations, as it would in a “closed world” situation.
to be explored as quickly as possible, so that the user’s true preferences can be determined.

### 6.10.1 Counting- vs. Decision-based Data

Because the goal of the reasoning process is to glean information as passively as possible, the main source of data is the amount of time for which various adaptations are invoked, as depicted for two users in Figure 6.16. However it cannot be assumed that the user is aware of all adaptations, nor that all possible, or at least preferred, adaptations are available for a given device. Therefore the invocation of an adaptation is not as informative as an active, closed, choice between adaptations. Such choices are rare, as Figure 6.15 implies. The level of inertia that the passively-collected data exerts on potentially fluctuating capabilities is controlled by how much historical data is used when computing preferences.
6.10.2 Usage-based Indicator Preference Deduction

The purpose of this section is to ascertain users’ preferences for different consequences of adaptations (i.e. TRIALS indicators and effects on other channels) based on the counted adaptation usage data most readily available. For each indicator and possible value in Table 6.2, a score is calculated that indicates how likely that value for that indicator will be at the top of the user’s usage ranking. The process for generating and testing this metric is detailed in section 8.5.

6.10.3 Adaptation Feedback and Control

So far, the only means of giving active feedback on adaptations that have been implied are: (implicit) acceptance and (explicit) rejection. It is asserted that invocation, implicit/explicit acceptance, rejection and “reconfiguration” are sufficient primitives for the reasoning process to learn from any adaptation. This not only provides for a simple workflow for users, but also makes it very easy for platform vendors to supplement any existing accessibility APIs with a minimal interface necessary to control adaptations.

Part of the rationale for this approach comes from the REpresentational State Transfer (REST) process developed by Fielding [32] for interfacing with web services. The key to the success of the approach is that instead of defining myriad APIs each tuned to dealing with a particular data schema (for example), the REST interface provides the primitives needed for discovering the structure of that data/service (as the results are returned in a structured form such as XML) and then subsequently manipulating (adding, updating or deleting values from) the service. If the field names and data types are known then Create, Read, Update and Delete are the only four primitives needed to deal with the data. REST combines those with discovery.

This approach is applied to adaptations: discovery is facilitated via associated capability metadata; consequences in information transfer and capability burden terms are understood from the TRIALS indicators and users’ acceptance or otherwise of these consequences informs the reasoning process about this and related adaptations. In the case where the user made a help request of the system, directed at a particular part/entity, and then chose the adaptation(s) from a list of possibilities, this can be considered a choice of one adaptation set above others (if it remains accepted).

The only further option, to “configure” an adaptation, is used to instruct the adaptation to provide configuration options to the user. Configuration options may be as simple as extent (as with a zoom adaptation) or be more elaborate (such as providing control over which type of colour perception deficit is corrected).
Figure 6.17: Abstract adaptation consequences. Feedback actions have implicit meaning and, thus, ramifications in terms of discerning users’ preferences.

adaptation would be, as usual required to notify the OS of changes it makes. These would be detected by the reasoner and capability data updated accordingly. The means by which configuration options are presented is not relevant here, though it would be ideal if a REST-like approach could be used in the sense of having a standard interchange format, such as ISO 24751 [64] technical adjustments, delivered using XML.

6.10.4 Expressiveness of Feedback

Simply invoking an adaptation when none was there previously, or removing it when it is the only one invoked does provide feedback in the sense that an adaptation was preferred (or not) to no adaptations. However the feedback can be considerably more expressive if these invoke/remove operations are carried out in the presence of other adaptations, as Figure 6.17 shows. For a detailed breakdown of the effects of these operations on abstract adaptations, in terms of TRIALS indicators and entity datatypes and modalities, see section D.4.
Figure 6.18: Generalisation of the adaptations decision tree given in Figure 3.11. Sets of adaptations (edges) are proposed to solve a problem (node), resulting in a new system state. Should the new state constitute a significant problem, the process is repeated. The consequences of accepting any given alternative (leaf) are recorded.

6.10.5 Decision-guided Adaptation Exploration

The limitations of usage-count data as the primary means of deducing user preferences can be overcome to a degree. As the system is used, genuine adaptation choices, as they occur, can be recorded. Should the user ask for adaptation suggestions (see subsection 3.5.3) then several candidate adaptation sets will be presented, giving the system an opportunity to suggest at least some sets in two key additional ways to the mappings above.

- Decision-based, if sufficient decision data are available and point to substantially different adaptations than the current usage-based data.
- If no decision data are available then a sensible fallback is to suggest sets that are some distance away within the TRIALS indicator space.

Naturally any suggestions are in the abstract domain and the reasoner would have to ensure that corresponding concrete adaptations were available to embody the requirements of the chosen abstract set.

6.11 Preference-directed Problem Resolution

When a problem is detected or declared (as in subsection 6.2.2), this problem must be mapped to sets of adaptations which may solve it. These adaptations, in turn, may introduce their own consequences, producing a decision tree such as that depicted in Figure 6.18, which is a generalisation of Figure 3.11, accounting for multiple adaptations being proposed at each stage.

The “problem” is either that user or T-device has insufficient capacity to transfer the required information for an entity (or whole channel). This may be caused
by increasing IE requirements, or decreased capacity on behalf of user or T-device. Each node represents the system state in TRIALS, user capability and IE levels—or, more likely, simply changes to the levels before the problem state, which are assumed to have been acceptable. The root node is the problem state; leaves are states in which the search terminated, either because no further abstract adaptations were found or all problems were solved.

1. Visit next problem node.

2. Suggest adaptation sets that could help (i.e. solve current problem, keep stability, improve redundancy).

3. Edges are those sets.

4. The next level of nodes represent the system state after each set of adaptations is applied.

5. These adaptations may have introduced problems of their own, which are explored in turn.

The stopping criteria are: all problems have been solved; or no further abstract adaptations are available; the original problem has been re-introduced in a child node (prevents infinite recursion); the same adaptation has been re-used in the same modality (again, prevents infinite recursion). It is also important to provide an opportunity to revisit the intermediate nodes, as these represent solutions that could be of merit. For instance: if a zoom is applied, the extra bandwidth requirement must be traded for time or information. Imagine that reflowing or just panning were used to allow the information to be accommodated at the expense of time. At this point, a workable solution is present, but it still contains one more “problem” in the raised time requirement. A Reduce adaptation may then be applied in order to remove information to free up time, thus degrading an already appropriate solution. This tree exploration differs from that in chapter 5 in that intermediate nodes are in fact valid solutions.

The structure of this process is similar to that of the main collaboration search from chapter 5 and it has the same properties of completeness. However the search space is dramatically smaller due to: (a) the fact that only a limited number of abstract adaptation sets can be used to solve a particular problem and only one problem is being solved at a time (as opposed to a whole set of entity assignments). User preferences in terms of TRIALS consequences are used to order the sets of potential abstract adaptations that are explored and provide a cost factor.
Figure 6.19: Adaptation at different levels: T-device (bottom) and mutation (top). The example shown is full-screen magnification at the T-device level, leading to two-dimensional panning, combined with a font-size increase mutation to some content, leading to additional one-dimensional scrolling.

### 6.12 Adaptation Conflicts

When a single user faces more than one simultaneous accessibility barrier, or a group of users share a T-device, the interference between adaptations may present further problems. This should be anticipated and mitigated, if possible. There are a number of levels, from coarse to fine, in which incompatibilities may present themselves.

**Gross capability/learning style** differences would be accommodated by the use of different T-devices for some, or all, entities and so do not result in conflicts, but may result in poor synchronisation.

- Severe sensory impairment.
- Strong data type preference (possibly due to learning difficulty).

**Burden** caused by the ancillary effects of an adaptation invoked for another user. The burden imposed depends on the devices in use (e.g., if an adaptation must be compensated for by scrolling, the scrolling burden and how it is perceived depends on the device used to afford this scrolling; one possibility was illustrated by Figure 6.9). For example: the effects of full-screen magnification: reduced viewport; extensive panning; increased cognitive load.

**Indicator preference** differences signify differing stances on the importance of various indicators (and their directions). A metric for assessing the degree of divergence is given in section 8.5.
CHAPTER 6. ADAPTATIONS AND CAPABILITIES

Table 6.10: Compatibility in T-device layer; i.e. adaptations applied to the device, or all entities rendered/input via the T-device.

<table>
<thead>
<tr>
<th></th>
<th>Aesthetic</th>
<th>Scroll (2D)</th>
<th>Replace (Type)</th>
<th>Replace (Mod.)</th>
<th>Remove</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aesthetic</td>
<td>≈</td>
<td>⋆</td>
<td>⋆</td>
<td>○</td>
<td>○</td>
<td>⋆</td>
</tr>
<tr>
<td>Scroll (2D)</td>
<td>⋆</td>
<td>≡</td>
<td>⋆</td>
<td>○</td>
<td>○</td>
<td>⋆</td>
</tr>
<tr>
<td>Replace (T)</td>
<td>⋆</td>
<td>⋆</td>
<td>≈</td>
<td>○</td>
<td>○</td>
<td>⋆</td>
</tr>
<tr>
<td>Replace (M)</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>≡</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td>Remove</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>≡</td>
<td>○</td>
</tr>
<tr>
<td>Share</td>
<td>⋆</td>
<td>⋆</td>
<td>⋆</td>
<td>○</td>
<td>○</td>
<td>≡</td>
</tr>
</tbody>
</table>

Symbols: ⋆ = compatible; ≡ = (self-)compatible (though extents may be varied); ≈ = potentially compatible (parameters other than extent may vary); ○ = incompatible.

Note: Replace (M), Reduce and Supplement (not shown) are equivalent as they take entities out of the current T-device and any channels between it and users.

Abstract adaptation preferences, within a particular (compatible) set of TRI-ALS preferences. For example: Folding vs. Panning.

Concrete adaptation preference, based upon the underlying abstract adaptation. In practice this is expected to be a minor concern, as each platform will necessarily have to have a different concrete version of the various abstract adaptations. It would likely only become significant when heavier ATs for people with recognised disabilities are introduced, as it might then encompass the differences in style between different screenreaders.7

Figure 6.19 depicts a situation that can occur; this may be tolerable to some users but not others. The process for resolving potential conflicts between users sharing the same T-device is exactly the same as that given in the previous section—conflicts at any of the layers above simply reduce the space through which the search travels. If the end results is not acceptable to any particular user, there are two options: either (1) re-seed the collaboration but with the shared T-device off-limits for the given user or (2) simply continue with the collaboration as-is. A slight modification of this would be to re-seed the collaboration search and stipulate that the user experiencing problems would receive a mirror of the shared display on a personal device, if it exists.

Tables 6.10 and 6.11 describe the compatibility of adaptations at the T-device and mutation (entity) levels. They are applicable when a single user employs multiple adaptations and/or when a T-device is shared between users. Should different users in the collaboration be using different T-devices then they may

7Until even larger ATs are made more mutable themselves.
Table 6.11: Compatibility in mutation layer; i.e. adaptations applied to one entity, or a group of entities, being rendered/input via a single T-device.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Aesthetic</td>
<td>≈</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>ə</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Fold</td>
<td>⋅</td>
<td>ə</td>
<td>ə</td>
<td>ə</td>
<td>ə</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Scroll (1D)</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>ə</td>
<td>ə</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Scroll (2D)</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>ə</td>
<td>⋅</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Replace (T)</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>ə</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Replace (M)</td>
<td>ə</td>
<td>ə</td>
<td>⋅</td>
<td>⋅</td>
<td>ə</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Reduce</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Remove</td>
<td>ə</td>
<td>ə</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
<tr>
<td>Share</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td>⋅</td>
<td></td>
<td>⋅</td>
<td>⊙</td>
<td>⋅</td>
</tr>
</tbody>
</table>

**Symbols:** ⋅ = compatible; ə = (self-)compatible (though extents may be varied); ≈ = potentially compatible (parameters other than extent may vary); ə = incompatible.

**Note:** Replace (M), Reduce and Supplement (not shown) are equivalent as they take entities out of the current T-device and any channels between it and users.

use “incompatible” adaptations—the effect of which would most likely reduce the degree of synchronisation. Finally, Table 6.12 gives an overview of situations in which conflicts may occur.

### 6.12.1 Modelling and Detection

A vast and growing range of concrete adaptations exists, as does an ostensibly infinite number of potential collaboration scenarios. Ideally, any incompatibilities between users’ required adaptations should be detected and mitigated before a collaboration solution is implemented by the system.

Two options exist for modelling incompatibilities based on preferences: (1) blacklist particular concrete adaptations that a user cannot tolerate, or (2) blacklist the parent abstract adaptations. Whilst option one could result in overloading users with interactions during collaboration set-up, option two could mis-classify concrete adaptations that are exceptions to the rule (i.e. discount particular concrete adaptations that may have been considered acceptable—false negatives), which could result in a poorer collaboration scenario (such as failing to accommodate all users on a local device, or failing to achieve the maximum information transfer to/from some users), or failure to reach a solution. This emphasises the assertion that being able to state reasons for failure (or means to improve a situation) in human terms could be of great use, as end-users may then be able to improve the situation through further manual intervention (whitelisting a given
Table 6.12: Potential conflict scenarios between users.

<table>
<thead>
<tr>
<th>Single-user?</th>
<th>Local vicinity?</th>
<th>Parent Device</th>
<th>T-device</th>
<th>Conflicts may occur between...</th>
<th>Areas that may be affected</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>⬤</td>
<td>⬤</td>
<td>Same</td>
<td>Same</td>
<td>Source and ancillary adaptations</td>
<td>Preferred TRIALS outcomes, source and ancillary channels</td>
<td>Fully-signed and vision-impaired person collaborating*; single-user whose needs suddenly change**</td>
</tr>
<tr>
<td>⬤</td>
<td>⬤</td>
<td>Same</td>
<td>Different</td>
<td>Ancillary adaptations, local communication</td>
<td>Ancillary channels, synchronisation</td>
<td>Sighted and blind people***</td>
</tr>
<tr>
<td>⬤</td>
<td>⬤</td>
<td>Different</td>
<td>–</td>
<td>Local communications</td>
<td>Synchronisation</td>
<td>People with quite different preferences or capabilities</td>
</tr>
<tr>
<td>⬤</td>
<td>⬤</td>
<td>Different</td>
<td>–</td>
<td>–</td>
<td>Synchronisation</td>
<td>People collaborating remotely</td>
</tr>
</tbody>
</table>

**Fields:** Single-user? = applicable to single-user scenarios?; Local vicinity? = Are all users in the same area (and therefore can communicate via non-electronic means)? **Symbols:** ⬤ = yes; ⬤ = no; – = not applicable.

* as they are sharing the same T-device, there is the potential for conflict between preferences as well as adaptations’ ancillary effects.

** this could be due to environmental changes.

*** If a suitable T-device were available, Braille output for the blind person would be preferred, to allow easier verbal communication between users.
concrete adaptation, in this case) if so inclined.

Reliable inference of adaptation incompatibilities could be of use in single-user as well as collaborative situations (one user may, for example, have multiple impairments). It will sometimes be possible to draw such inferences from a user's capabilities. For example; if a person has poor motor dexterity, it is quite likely that full-screen magnification, which introduces a high scrolling requirement, would be inappropriate—the link between the two was illustrated by Figure 6.9. If someone with motor control difficulties were to collaborate with a vision-impaired person, both parties may find a larger GUI widget size mutation (and folding, as in Figure 6.7) to be both beneficial and highly compatible.

6.12.2 Search Behaviour Remarks

The search for an alternative adaptation need not be carried out in brute force fashion. From the requirements of each user involved in the collaboration, the choice of adaptation may be constrained effectively.

**Vision and motor problems.** Full-screen magnification is suitable for vision problems but not motor difficulties, due to the increased scrolling requirement. However the application of a system-wide “abstract entity enlargement” mutation, in combination with a fold adaptation, would satisfy both impairments.

*As the problem tree (Figure 6.18) is explored in a complete fashion, the abstract entity option will be found—but will only be applicable if the host OS supports such mutations.*

**Cognitive and vision problems.** If a novice computer user, who is learning how to send emails, wishes to be taught by a vision-impaired person, then an adaptation may be required. Scrolling would likely confuse a novice user, as the system would look markedly different whilst they were learning to how it would appear in their personal daily use. The same would be true of the folding solution employed above. However, a “reduce” applied to the interface as a whole could provide sufficient space for an entity-enlargement mutation to be successful. Whilst the interface would now be more minimal, this may actually be of help to the novice user, as there would be less cognitive load involved in learning to use the system.

*This collaboration would fail if the “teacher” had a preference against the “reduce” adaptation due to being an expert user. However, it would be trivial (and automatic human social behaviour in this case) for the vision-impaired person to tolerate this adaptation, as it is for the learner’s benefit.*
If such an alternative does not exist, this can be noted. The overall collaboration search will then proceed to the next-best set of devices. If, ultimately, no solution is found, the reason for failure to find an alternative adaptation at this stage can be communicated to the user, so that in future a resolution may be possible. (Ultimately, if no local collaboration solution is found then a remote one is likely to be possible. The hypothesis here, being tested by Sus-IT, is that most people would prefer “error messages” in terms of human capabilities rather than technical terms.)

### 6.13 Context

Context encompasses all adaptation-related activities, so this includes both preference deduction avenues pursued in subsection 6.10.2 and subsections 6.10.3 and 6.10.5.

The user preferences observed so far could be taken to apply to the whole course of interaction or, perhaps more usefully, just to certain usage scenarios (e.g. certain device types, data types, application types). Examples may include “I prefer to use TTS for reading web pages on mobile devices”; “I prefer to disen-
gage full-screen magnification when using full-screen media players” and “When supported by the platform and applications, I prefer folding the interface to full-screen magnification”.

The task of defining “context” in any given domain, as well as measuring it, is non-trivial, as has been discussed in the literature review. Inappropriate use of context to direct decision-making can lead to unpredictable interactions (from the user’s perspective) in both adaptive and recognition-based interfaces. However the current work focuses on abstract adaptation consequences and is based on knowledge of the user generated in large part by monitoring the user’s own actions. It is proposed that the knowledge gleaned is sufficiently fundamental to be reliable and perhaps applicable to new situations.

Specifying a context would involve specifying zero or one elements from each of the following sets of information, which could be seen as providing a set of “tags” for the adaptation preferences.

- Whether the adaptation is to content or UI.
- Whether the channel supports sharing (e.g. windowing, on two-dimensional T-devices) of multiple applications simultaneously and, if so, whether one application is currently being rendered full-screen.
- The general nature of the application (e.g. passive or active interaction).
- The type of device being used. It is envisaged that this may include “desktop” and “mobile”.

Specifying a given context, then, involves selecting an element from the power set of those given above; e.g. \( \{V, UI, \ldots\} \). The granularity of the preference is determined by how many context elements are specified, allowing very coarse or fine changes to the preference order to be specified. Most of these sets have fixed and small cardinality. The “tags” in others (such as device type) are expected to change, slowly, over time, but still be low in number (relative to the set of possible concrete adaptations) at any given time. A further category of tag could be added to allow users to link certain preferences to situations (such as “home” or “work”).

Now that context has been introduced, it is possible to construct the mapping from concrete to abstract adaptations, introduced above. This would consist of an ordered list of concrete adaptations that had been applied in any given context (including the default). The adaptations proposed to the user, should help

---

8As web-based information services become increasingly popular and have exposed the public to the benefits of “tagging” this seems to be compatible with good usability.
Table 6.13: Example contextual descriptors.

<table>
<thead>
<tr>
<th>Category</th>
<th>Example values</th>
<th>Useful for differentiating (e.g.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modality</td>
<td>C, V, A, M</td>
<td>Fundamental data types</td>
</tr>
<tr>
<td>Scope</td>
<td>UI, Content</td>
<td>Web pages/documents vs. application</td>
</tr>
<tr>
<td>Interaction</td>
<td>Active, Passive</td>
<td>Document editing vs. media-playing</td>
</tr>
<tr>
<td>Sharing</td>
<td>Viewport, Full-device</td>
<td>Windowed vs. full-screen</td>
</tr>
<tr>
<td>Location</td>
<td>Home, Work, (Other)</td>
<td>Role-based adaptation preferences such as consumer (home) vs. creator (work)</td>
</tr>
</tbody>
</table>

be requested, would be formed from those indicated by the user’s capabilities and preferences in terms of adaptation consequences, inferred from the preference distributions discussed in section 6.10 and depicted in Figure 6.16).  

6.13.1 Reflecting Users’ Views on Context

As above, users may prefer different adaptation consequences in different situations. Further, they may also differ on how they demarcate usage contexts, as illustrated in Figure 6.22. A technique is required to detect and reflect a user’s preferred view of context. As each possible usage context is associated with a distribution of abstract adaptations, it must be possible to determine the similarity between these distributions. This can be achieved simply by comparing the indicator scores from section 8.5.

When a user makes an adaptation, it is applied in a given context. Imagine that all of the available contextual tags are applied, giving the label \{V, UI, home, web\}. The same person may also have a set of adaptations labelled \{V, UI, work, web\}. If the distributions are both the same or very close, then, as they are only one element different, they could be seen more generally as \{V, UI, web\}, assuming there was no existing and different distribution with that label. Should the user in fact have the same preferences for any UI elements, regardless of application, the more general label \{V, UI\} may ultimately be adopted and be in contrast to, for example, \{V, Content\}.

In time, the user acquires a new device, a portable music player, which works by voice control and output only. As the device has \{A, UI\} and \{A, Content\} labels, and the most similar in the user’s preferences are the corresponding visual distributions, these could be used as a starting point for the new device. At this point, it is possible that different interaction styles will result in a preference to treat aurally-rendered UI and content differently to their visual counterparts—and more consistently amongst themselves. Alternatively, the UI/Content split may continue. If any further consolidation of preferences were to occur (which is by
Figure 6.22: Different users’ views of contexts and their potential combinations. In the interests of minimising cognitive load, the fewest distinct zones of adaptation must be achieved that satisfy both users.
no means inevitable), then the two discussed outcomes would be \(\{V\}, \{A\}\) and 
\(\{UI\}, \{Content\}\) respectively.

The general process of grouping similar data-points—in this case by adaptation preference similarity—is known as clustering and may be carried out hierarchically, as above (the consolidation of the labels is a simple set intersection at each clustering stage). Unfortunately in the general case the algorithm for accreting clusters has time complexity of \(O(n^3)\), though an \(O(n^2)\) approach may be used in cases when cluster distance may be measured from the edges of clusters [98]. There is a potentially large amount of potential contexts (up to \(2^{\text{tags}}\), but there would only be one element in each “cluster” and the algorithm (which is greedy by nature and continuously combines the nearest clusters) would be stopped from combining distribution/context points that were more than a tiny distance apart.

6.14 Summary

This chapter has refined and developed the reasoning process in order to accommodate more subtle adaptations, of relevance to scenarios involving minor-to-moderate impairments, as follows.

- A model that can express both contemporary ATs and more fine-grained adaptations from research and that are anticipated to become more popular as pervasive computing expands.

- Expression of the effects of adaptations in terms of information transfer, in such a way that adaptations can be reasoned about in an abstract manner. This allows the reasoning process to accommodate existing and future adaptations.

- Reasoning regarding multiple concurrent accessibility barriers and impairments and methods for detecting and addressing the interference between adaptations (as perceived by individuals).

- Processes to support collaboration in more realistic situations—and to find suitable alternatives (should they exist) in case of conflicts.

The final chapter on reasoning follows, in which some of the key challenges of temporal adaptivity reasoning are explored.
Chapter 7

Exploring Temporal Dynamism and Stability

This chapter calls attention to some of the challenges in extending the reasoning process developed so far into the temporal dimension.

One way to deal with time is to effectively "snapshot" the system at various points and have the reasoning process interpolate between them. Another way would be to restart from scratch and create all structure in a form that is aware of time and its potential effects. This approach was used in creating formal models for analysing multimodal interfaces [26, 68].

7.1 Inference

It is desired that the reasoning process be able to make some simple, robust, inferences regarding users’ adaptation behaviours. Situations in which these inferences could assist people are plentiful: the person that “always” zooms in on wordprocessor documents after they have been opened (due to the default level being inappropriate on a given screen or OS and no suitable abstract adaptation available to have corrected this); the vision-impaired user of a screen magnifier who chooses to disable the magnification in certain full-screen applications (e.g. video players) in order to see an overview of the content.

These two behaviours are non-temporal in nature and would be learnt reliably by the context analysing process (section 6.13). However there may be some behaviours related more to time than to any sort of context the system can detect. Many users (particularly older and/or disabled people) tire quickly over time and, thus, adaptations could be brought in to assist the user as the session progresses—and “reset” for the beginning of the next session.

Time series analysis, which analyses patterns over various timescales (e.g. a
day, week, month, year). For example, it highlighted that during certain television
programmes, additional electricity generation must be brought online in order to
cope with the demand for increased kettle usage during commercial breaks.¹ This
is not an unrelated example—the use of data collected over long time periods or
across multiple people shows that robustness of inferred rules can be sufficiently
high.

7.2 Expressing Changes in Capabilities and
Preferences

The previous chapter discussed the refinement of capability estimates, abstract
adaptation preferences and T-device orderings via a feedback loop. These values
will, of course, fluctuate over time, possibly over the course of a session or longer
periods. The sensitivity of the system to change depends on the size of window
used; too small and the system will react too quickly, but too large and the weight
of historical data will render the system insensitive.

Returning to the example of a user tiring over the course of a session, it is
possible to detect that the user will prefer to read large amounts of text from a
display initially, but by the end of the session, the same person will prefer text
to be rendered aurally. In order for the reasoner to ascertain the appropriate
adaptation to afford this change, the situation must be modelled. A logical way
to do this would be to update the order of preference of different types of T-device
for a given datatype.

If the initial set-up process were to be re-run with this changed data, then a
different rendering would have resulted, in which certain entities would have been
assigned to different T-devices. By comparing the different assignments of entities
to T-devices in the anticipated future rendering, the adaptations (replacements,
in this case) required to convert the existing rendering into the desired one can be
determined.

7.3 Expressing the Effects of Temporal Factors
in Information Transfer Terms

In the previous chapter, it was asserted that channels’ capabilities may not be
arbitrarily improved. This is true, however these capabilities can change. Con-
tinuing the example of the tired reader above, the person’s visual utility could be

¹http://en.wikipedia.org/wiki/TV_pickup
Table 7.1: Example trade-offs between battery life and capabilities.

<table>
<thead>
<tr>
<th>T-device</th>
<th>Modification</th>
<th>Power saving</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screen</td>
<td>Brightness</td>
<td>1% per gradation</td>
<td>Brightness; contrast</td>
</tr>
<tr>
<td>Network</td>
<td>Disable 3G</td>
<td>4% on average</td>
<td>Network bandwidth</td>
</tr>
<tr>
<td>Speakers</td>
<td>Disable</td>
<td>10% on average for app.</td>
<td>Disables channel</td>
</tr>
</tbody>
</table>

**Note:** the table illustrates a purposely simplistic scenario in which for each level of each adaptation, a particular level of power is saved. However, the approach could easily be adapted for realistic scenarios, dependent on information from device manufacturers.

modelled as falling over the time of the session. This, in turn, would limit the capacity of the channel, to the point where an adaptation would be required in order to allow the user to continue reading comfortably (or at all). Whilst it would not be possible to artificially boost the user’s visual acuity, it is clearly possible for the capability to fluctuate. This fluctuation could be tracked over the course of each session.

A similar effect to that of tiredness applies to portable devices: that of battery life. This can be modelled in the same way, which presents some interesting opportunities for the reasoning process. It is known that battery life is limited. However, there are actions that can be taken to prolong battery life: most prominent amongst these would be to lower display brightness and, after this, avoid using costly (power-wise) network connections such as 3G. All such actions would constitute modifications to the T-devices of a given parent device (such as a smartphone). Again: the reasoner has no concept of the cause, but the effects can be modelled, assuming that the device manufacturer provides information linking these actions to their effects on channel capabilities.

The trade-offs in real-world scenarios are likely to be considerably more interlinked and complex; the table is simply illustrative. It is acknowledged, for example, that the power savings per gradation of a given adaptation are unlikely to be linear (this can be addressed by using a correct scale, or providing tables of power savings for each modification in order to calibrate the calculation). In order to ascertain the best values for these trade-offs at any given time, based on the capability requirements of the user and demands from the environment and entities being presented or input, techniques such as linear programing may be used to express and solve the power-bandwidth trade-offs. This presents an exciting opportunity for devices to become more efficient by tuning themselves to the capability requirements of any given situation.
Figure 7.1: Stability problem. The Xs represent adaptations being applied to the system. The goal is to accommodate the unexpected (middle) adaptation in a way that minimises the disruption of the forthcoming predicted adaptation. The blue lines highlight the relative amount of time a sub-optimal adaptation (if applied in order to improve stability) would have an effect on the system.

7.4 Stability

Stability is the resistance of a given interface or content rendering to change over time. Whilst most of this work has advocated adaptivity as a vehicle for convenient and timely change, largely before rendering an interface, in order to overcome accessibility barriers, it is important to temper this with the fact that, as has been pointed out, people do not like change—unless the perceived benefits markedly outweigh the perceived costs. In order to both: (1) keep renderings stable—and therefore usable—over time and (2) to accommodate fluctuating capabilities and reactionary adaptations, the notion of stability is formally introduced. Figure 7.1 provides a depiction of the fundamental stability problem.

There are three possible courses of action in this situation, as follows.

- Have the three adaptations be invoked separately (causing three potential instances of disruption overall).

- Make minor changes to the current rendering in line with the unexpected adaptation (sufficient to solve the problem, but not terribly different to the current rendering due to the expected future adaptation).

- Make more substantial changes to the current rendering, in line with the expected adaptation (causing more disruption now than would have been the case, but less disruption overall, due to accounting for the future adaptation).

The decision will be influenced by the following factors.

**Cause.** What was the initial trigger for the adaptation? This could either be the user or the environment.

**Severity.** Action will be taken dependent on the relative strengths of the unexpected and predicted adaptations. However these adaptations could affect different channels (even different modalities). Therefore “severity” here
means the relative strengths of the adaptations in any channels that they both affect. (If neither adaptation affects remotely the same channels, then there is no potential stability conflict.)

Note that the most severe adaptations to a channel are: replacement of content within the channel and replacement of the channel (i.e. a switch to another T-device and therefore possibly modality).

Some basic rules that constrain the decision-making process are needed, to ensure that sensible—effectively failsafe—decisions are reached.

- The cause of the unexpected adaptation.
  - If caused by the environment, then it should take precedence (i.e. if there is a need to switch modality imposed by the environment, then this simply must be implemented).

- The severity of the unexpected adaptation relative to the predicted adaptation.
  - If the unexpected adaptation is more disruptive than the predicted adaptation, then it should be adhered to exactly (as it is either caused by the environment, as above, or the user, who is clearly unhappy with the present and expected situations).
  - If, however, the expected adaptation is the more severe, the choice remains as to whether to align the current rendering closer to it, or closer to the current situation. In these cases, past user behaviour, as well as preferences regarding system intervention, would be consulted.

### 7.5 Accessibility–Stability–Device Compromise

Continuing the example of the tired reader brings to light an important compromise that must often be made when an unforeseen problem is detected or declared: that between keeping the existing interface stable; improving its accessibility and honouring T-device constraints such as the desire to preserve battery life.

### 7.6 Summary

This chapter has highlighted the key matters of temporal reasoning that a real-world CAAR process must address.
CHAPTER 7. EXPLORING TEMPORAL DYNAMISM AND STABILITY

• Means to detect temporal patterns in capability fluctuations—particularly those that are related to times alone; examples include daily routine but also, and perhaps more importantly, the time relative to the start of the session.

• Methods for modelling the influence of external effects on channel capacities.

• A capability-focused approach to T-device constraints such as power management.

• Managing the stability–accessibility–device balance over time.

In the following chapter, the reasoning process as a whole is tested and evaluated technically.
Chapter 8

Analysis

Technical justification and mechanical testing of key elements of the CAAR process developed is undertaken in order to assess how well it meets the technical requirements. Section 8.6 discusses some key technical factors in how well this work complements existing research and industrial artefacts (and may complement future systems).

A key element of the design and function of the CAAR process is that, as it builds upon already-existing adaptations, it will benefit from the testing and rigour that has gone into the development of those adaptations. The goal, therefore, is not to prove that those individual adaptations work; rather that the reasoning process that controls the application of adaptations works. This means testing that it can react appropriately to basic inputs as well as having the potential to learn each user’s perceptions of the interference between adaptations—and take appropriate mitigating action.

8.1 Requirements

The key goals for the thesis were set out in the literature review.

- **Reasoning about accessibility, in terms of DUET constraints**—thus mapping users with minor-to-moderate impairments to appropriate accessibility adaptations.

- **Enabling “good-enough” bootstrapping on new devices or in new situations, based on users’ known capability (and underlying) preferences.**

- **Operating as passively as possible, allowing users to benefit either from improved adaptability or more active adaptivity, at their choice.**

\(^1\)Tests were run on an iMac with a [64-bit] 2.93GHz Core 2 Duo processor (the processes were largely CPU-bound; memory consumption was negligible).
• Supporting accessible ubiquity and collaboration (particularly with respect to output rendering on shared devices).

• Providing a unified method for communicating to the adaptivity system; minimising interaction with ATs and maximising interaction with the application’s interface—both for end-users and, equally, platform vendors and developers.

The following sections examine several sets of related goals in turn.

8.1.1 Inherent to the CAAR Process

The reasoning process developed was designed with the fact that barriers may be caused by more than impairments on behalf of the user in mind. Environmental and device-related factors are acknowledged as potential causes of accessibility problems (addressing Goal 2). Further, the reasoner assumes a complementary role to any adaptations that users may wish to engage (perhaps undetectably from the system’s perspective); users can decide how much control the reasoner has and this can be tuned during use (addressing Goal 5).

Finally, as adaptations are dealt with in a unified way, with reasoning being in terms of their effects on (i.e. changes to) information transfer between the user and system, they are all handled consistently. Users are offered the same opportunities for feedback on all adaptations. Thus the technical aspects of Goal 6 have been addressed. Interaction workflows and UIs are under development by Sus-IT presently to afford users this control and assess its performance. Some very early ideas in this area are presented in subsection 3.5.3.

8.2 Classification, Indicators and Possible Adaptations

The TRIALS indicators and abstract adaptation types, used for classifying adaptations and major user preferences, must fulfil the following goals.

1. The dimensions of the space should be orthogonal.

2. They must allow all possible adaptations relevant to CAAR to be uniquely identified.

In accordance with the separation of concerns, adaptations (and their consequences) are classified in a number of ways, as follows.
Figure 8.1: How the global indicators are determined. Time, Space and Information volume are orthogonal, as are Aesthetics and Layout. Global indicators are aggregated across all channels.

**Indicators.** The TRIALS indicators represent the consequences that can occur from invoking a given adaptation.

**Abstract type.** There are a number of different types of abstract adaptation. Some have the same TRIALS consequences, but work using different methods. There are some atomic adaptations (e.g. Reduce/Expand) and some composite ones (such as Fold).

**Data types.** As this work is regarding data-driven applications, a key additional facet of classification is the effects of an adaptation on data type.

The primary method of classifying adaptations used by this work is the combination of TRIALS consequences and abstract adaptation type (subsection 6.6.3). However, underlying this there is a more fundamental space, defined by a series of orthogonal properties, as depicted in Figure 8.1.

### 8.2.1 Indicators from Orthogonal Facets

For a problem-space description to be robust, its fundamental dimensions must be orthogonal (i.e. non-interfering). Further, the universal space of adaptations cannot necessarily be modelled; this work acknowledges, but does not attempt to completely address, cognitive adaptations. This leaves the realm of adaptations
which have some effect on the “physical” information transfer process—i.e. those concerned with sending a given (logical/mechanical) amount of data from one party to another, via a channel.

The following pairs of indicators are considered to be orthogonal and may be considered as independent axes. They are relevant at different layers within the system (collection of channels; specific channel; sub-channel application parts and entities).

**Entity/Part** indicators.

- Information volume : share of part/channel bandwidth $\rightarrow$ time required to render.

  For $n$-dimensional T-devices rendering or inputting $n$-dimensional temporal information.

- Information volume : share of part/channel space $\rightarrow$ space required to render.

  For $n$-dimensional T-devices rendering or inputting $\{1 \ldots n\}$-dimensional information. Should insufficient space be available, scrolling may be used (increasing consumption time) or the information may be reduced, removed or replaced.

**Entity/Part or Channel** indicators.

- Aesthetic transform : layout changes $\rightarrow$ similarity measure.

  Whilst these do not have defined units, they are deemed to be orthogonal within this adaptation classification. The combination of different magnitudes of aesthetic and/or layout transformation could be seen as a measure of similarity, as depicted in Figure 8.2.

**Channel** indicators.

- Information volume : bandwidth $\rightarrow$ time required to render.

  The product (area under the graph) gives required rendering time (or space).

- Space requirement : space available $\rightarrow$ space shortfall.

  This would be addressed by using adaptations to introduce scrolling, remove/reduce information or replace the affected entities with (physically/logically smaller) alternatives.

From these, the global (across all channels) indicators can be derived.
• Global information volume is the sum of that presented by each entity, part and channel.

• Global information consumption time is the maximum of the sum of the consumption time required by each entity, part and channel.

• Redundancy increases when information is duplicated across channels (even, or particularly, in different modalities).

The above pairs of facets are orthogonal as long as the following conditions hold.

**Aesthetics and layout are independent.** This is ensured by defining layout changes as those that change the relative (physical or temporal) positions of entities. Aesthetic transformations are those changes that may assist cognition or the fitness of the system for its environment (e.g. screen brightness). Changing the datatype used to represent an entity may have cognitive implications but is not purely aesthetic. Examples include describing the time in pictorial form, or perhaps using word phrases instead of numerals.

**Aesthetics and layout are independent of bandwidth.** Should more space become available (which would occur following a bandwidth share adaptation), a more spacious layout may be adopted. This could be useful for people with motor impairments (requiring large widgets), or those who prefer more “white space” to aid cognition. However, it is possible to change the layout of information without altering the bandwidth (i.e. space or time within a channel) required by it (see subsection 6.6.3; the adjustment of whitespace).

### 8.2.2 Comparing Aesthetics and Layout

Aesthetics and layout have been purposely left vaguely-defined for several reasons: (1) they are highly related to cognition, which is under-researched but out of the
present scope; (2) they are, as a result, highly subjective and, in turn, (3) quite domain-specific.

There are many metrics, guidelines and techniques used to measure and/or judge aesthetics (including, for example, the golden ratio). As the present work aims to be as domain-agnostic as possible, these methods are not prescribed, though a range of possibilities exist. In order to compare aesthetics and layout for similarity, for example, metrics ranging from the simple aspect ratio to Gajos’ similarity measures for cross-platform GUIs [45].

### 8.3 Collaboration Set-up

The test scenario is that from section 5.2. The algorithms from that chapter were coded in Python (a dynamically-typed, bytecode-interpreting scripting language). The code was not heavily optimised, largely because the main measure of performance for branch-and-bound searching is by how much it reduces the search space—if the technique finds the optimum solutions quickly in that sense, it is worth investing the effort into optimisation.

The third user added to some collaborations does not, unlike the first two, bring any personal devices. Also the third user can access all modalities. The task was to set up a shared collaboration with the same application as in section 5.2. To assess the performance with a more realistic, complex, application, the temperature converter was scaled up to be four times the original size in terms of number of entities.

Constraints in use: removed part fragmentation; ensured all entities were shared; ensured there were no unreachable shared entities. The constraint to remove assignments involving users’ personal devices was toggled (as shown in the results). Cost factors used sought to: minimise the number of non-optimal entity-T-device assignments as well as minimising those with weak (or worse) coherence.

The algorithm employed ordering functions that visited higher-dimension T-devices first for users with the visual modality and lower-dimension T-devices otherwise.

### 8.3.1 Complete Search Performance

Table 8.1 gives some results from the complete search. As can be seen, only a very small fraction of the search space is explored in order to find the optimum solution. Further, when adding additional users who do not bring their own personal devices—or, as in this case, no cost factors requiring additional cross-modality redundancy are in place—no further nodes need be explored. There is naturally
CHAPTER 8. ANALYSIS

Table 8.1: Complete Search Performance: how much of the search space must be explored before the optimal solution is found and how long does this take?

<table>
<thead>
<tr>
<th>U</th>
<th>P</th>
<th>E</th>
<th>Solutions</th>
<th>Examined</th>
<th>Best Node</th>
<th>Time</th>
<th>Constrained</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>7</td>
<td>$1.5 \times 10^{23}$</td>
<td>14,337</td>
<td>8</td>
<td>1s</td>
<td>1s</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>84</td>
<td>$1.4 \times 10^{278}$</td>
<td>172,033</td>
<td>85</td>
<td>158s</td>
<td>62s</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7</td>
<td>$1.5 \times 10^{23}$</td>
<td>14,337</td>
<td>8</td>
<td>2s</td>
<td>1s</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>84</td>
<td>$1.4 \times 10^{278}$</td>
<td>172,033</td>
<td>85</td>
<td>215s</td>
<td>70s</td>
</tr>
</tbody>
</table>

**Fields:** U = users; P = parts; E = entities; Constrained = time taken when solutions were constrained to not include users’ personal devices.

Table 8.2: Seeded Search Performance: how much of the search space must be explored before the optimal solution is found and how long does this take?

<table>
<thead>
<tr>
<th>U</th>
<th>P</th>
<th>E</th>
<th>Solutions</th>
<th>Examined</th>
<th>Best Node</th>
<th>Time</th>
<th>Constrained</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>12</td>
<td>84</td>
<td>$1.4 \times 10^{278}$</td>
<td>21,505</td>
<td>85</td>
<td>25s</td>
<td>10s</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>84</td>
<td>$1.4 \times 10^{278}$</td>
<td>21,505</td>
<td>85</td>
<td>34s</td>
<td>12s</td>
</tr>
</tbody>
</table>

**Fields:** U = users; P = parts; E = entities; Constrained = time taken when solutions were constrained to not include users’ personal devices.

Extra processing to do with respect to any shared devices, but it appears to only increase time complexity linearly.

Further, the model of assigning entities to devices fits in well with existing adaptive systems such as SUPPLE, as this is the starting point of their reasoning processes.

Whilst this search is clearly not suited to real-time use, a number of code optimisations can now be justified and would boost performance dramatically. These include: incrementally computing the cost of each partial solution as entities are added to it; caching results regarding the T-devices users can access and results of calls to the ordering function.

### 8.3.2 Seeded Search Performance

The algorithms were slightly modified to filter the power sets of available T-devices as in section 5.6 and the tests were repeated with the solution already providing a Screen for output and Keyboard and Mouse for input. The results are given in Table 8.2. Clearly this approach further dramatically reduces the search space and appears that it could easily be used in real-time if the above optimisations were enacted.
8.3.3 Capability-based Search

The preferences for certain datatypes to be rendered on certain device types, deduced in section 6.10, are applied to the search by way of T-device-ordering functions specific to a given user.

8.3.4 Extension: Pareto Frontier

Due to the nature of the current search process, it is necessary to tune the constraints and cost factors in order to avoid searching for solutions with potentially conflicting goals. The current cost factor checking process treats each factor separately, but does not regard a solution as better if it trades advancement in one factor for a retrograde step in another.

Ideally, one could imagine a search process that would make all of these trade-offs and yield the set of optimal solutions that result. This set of solutions would constitute the Pareto frontier—all possible non-dominated solutions given the underlying cost factors. A number of methods have been developed for integrating Pareto optimality with branch-and-bound [103], as well as some less exact but potentially much faster approaches based upon genetic algorithms [27].

These avenues may be worth exploring in future to provide fallbacks in complex multi-user situations, however it is clear that the techniques developed here represent a solid foundation—and it is very important to note that implicit direction from users (i.e. which devices they initially choose to use) affords dramatic problem space reductions.

8.4 Mappings Involving Capabilities and Adaptations

The goals of this area of the overall CAAR process, depicted in Figure 8.3 and described in section 6.8, are as follows.

- To build up a picture of pertinent capabilities, based on adaptation invocations in a given context (and other sources of data, if available).
- To suggest adaptations based on capabilities.
- To suggest alternative adaptations, of which the user may be unaware.
- To trigger appropriate adaptations (or, in the case of specialist adaptations such as compensation for particular motor problems, capability tests) when certain capability levels or changes are detected.
In conjunction with users’ preferences (see section 8.5), these suggestions allow users to both explore viable alternative adaptations and refine existing adaptations. Figure 8.4 depicts the mapping processes from the user’s perspective. It does not include matters relating to feedback on adaptations and abstract preferences; these were dealt with in subsection 6.10.3.\footnote{In summary: entity–channel assignments do take effect immediately and will affect future renderings; general adaptation feedback is logged and will thus influence future indicator preferences.}

### 8.4.1 Approach to Accuracy

The mapping method described in section 6.9 adopts the Monte Carlo method, in which a value of interest, often relating to a complex physical system, is closely estimated through the use of aggregated random sampling. The steps involved in using the method are: (1) enumerating the possible inputs; (2) selecting inputs
randomly; (3) using an input in a (deterministic) calculation and (4) aggregating results to form the estimate of the quantity in question. The stages in terms of capability mapping are as follows.

1. Enumerating all possible adaptations (i.e. those available for the current T-device).

2. Observing the usage of adaptations by the user as the random variable, over time.

3. Using the trivial mapping to relate usage of an individual adaptation to the implicated capabilities. The result of one unit of usage of one adaptation is effectively a list of capabilities.

4. Aggregating the results (using weightings, as explained) to create a ranked list of the most-implicated capabilities.

The results of the process as described do not indicate expected ranges of values for the capabilities concerned—however the technique is extended to do this in the following sections, with no increase in computational complexity (and limited increase in practical resource requirement).

8.4.2 Appropriateness of Capability Mapping and Adaptation Suggestions

In order to discover if the mapping and scoring technique is useful and robust, a simulation was carried out. A list of capabilities and a mapping from adaptations to affected capabilities (see Appendix E) was set up. The simulation was then carried out as follows.

1. A user was simulated by picking \( c \) capabilities from the list. Each capability was assigned an importances in the range \( 0 \leq i \leq 1 \) at random, so as to give an ordered list of capabilities.

2. Invocation of the adaptations was simulated by simply incrementing a counter for each adaptation that was linked to one of the user’s capabilities. This produces a ranked list of adaptations.

Steps up to this point would have been carried out by the user in real-world usage; the system would not have access to the accurate list of users’ problem capabilities.
3. The corresponding capabilities were inferred by reverse lookup on the mapping of adaptations to capabilities. This produces a ranked list of capabilities (as a capability may be touched by multiple adaptations).

Because many adaptations can affect multiple capabilities, but the user may only be having difficulty with a subset of these, it is at this point that spurious capabilities can be introduced. The system’s job, therefore, is to resist this “capability creep” by cross-referencing with other adaptations.

The following statistics were collected.

- What fraction of the original problem capabilities have been identified?
- What fraction of the original problem capabilities have not been identified?
- How many spurious capabilities have been identified? (This was recorded as a fraction of the length of the problem capabilities list.)
- What fraction of the spurious capabilities are in the top half of the inferred capabilities list? (This was an attempt to ascertain how vulnerable the approach might be to interference.)

4. A ranked list of adaptations was then inferred from these capabilities using the technique from algorithm 6.1. The same statistics were recorded for this list.

These steps were run 1 and 10,000 times on the hand-crafted mapping that was created—though it became apparent that the results converged at around 1,000 runs, so that figure was used in future. However, the results would certainly be peculiar to that mapping and thus, to better understand how the method performs in different scenarios, a range of mappings were automatically generated using the algorithm 8.1 and the simulation run on them also.

The results of all of the simulations are given in Table 8.3. As would be expected, many more capabilities than those that could be causing problems are implicated when mapping from adaptations to capabilities. This results in a large number of “intruder” capabilities in the top half of the ranked capabilities list. However, when the mapping (in this case back) to adaptations is made, it appears that this “intruder rate” declines to an acceptable level.

A further run of simulations was carried out, as in Table 8.4. In this set of simulations, one quarter of the adaptations initially “used” by the user were removed at random from the usage logs. The goal was to see if they were then suggested. For each simulation run, all of the removed adaptations had to be suggested in order to count as success. The table shows that the mapping approach, while simple, appears to be effective in suggesting appropriate adaptations.
Table 8.3: Results of mapping simulations in normal mode.

<table>
<thead>
<tr>
<th>Range</th>
<th>Runs</th>
<th>Caps. per adaptation coverage</th>
<th>Capability Included</th>
<th>Capability Excluded</th>
<th>Capability Spurious</th>
<th>Top-half Included</th>
<th>Top-half Excluded</th>
<th>Spurious Included</th>
<th>Top-half Spurious</th>
<th>Top-half</th>
</tr>
</thead>
<tbody>
<tr>
<td>0–2</td>
<td>1</td>
<td>1.30</td>
<td>0.38</td>
<td>0.60</td>
<td>0.40</td>
<td>0.10</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1.30</td>
<td>0.38</td>
<td>0.38</td>
<td>0.62</td>
<td>0.18</td>
<td>0.13</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>1–3</td>
<td>1</td>
<td>1.50</td>
<td>0.41</td>
<td>0.40</td>
<td>0.60</td>
<td>0.20</td>
<td>0.17</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>1.50</td>
<td>0.41</td>
<td>0.41</td>
<td>0.59</td>
<td>0.21</td>
<td>0.13</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>2–4</td>
<td>1</td>
<td>2.80</td>
<td>0.59</td>
<td>0.60</td>
<td>0.40</td>
<td>0.70</td>
<td>0.15</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>2.80</td>
<td>0.59</td>
<td>0.58</td>
<td>0.42</td>
<td>0.79</td>
<td>0.23</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>3–5</td>
<td>1</td>
<td>4.10</td>
<td>0.76</td>
<td>0.60</td>
<td>0.40</td>
<td>1.30</td>
<td>0.42</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>4.10</td>
<td>0.76</td>
<td>0.76</td>
<td>0.24</td>
<td>1.26</td>
<td>0.27</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>4–6</td>
<td>1</td>
<td>4.90</td>
<td>0.79</td>
<td>0.80</td>
<td>0.20</td>
<td>1.30</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>4.90</td>
<td>0.79</td>
<td>0.80</td>
<td>0.20</td>
<td>1.41</td>
<td>0.28</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>5–7</td>
<td>1</td>
<td>6.00</td>
<td>0.83</td>
<td>0.90</td>
<td>0.10</td>
<td>1.50</td>
<td>0.17</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>6.00</td>
<td>0.83</td>
<td>0.83</td>
<td>0.17</td>
<td>1.53</td>
<td>0.28</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>6–8</td>
<td>1</td>
<td>6.90</td>
<td>0.93</td>
<td>0.90</td>
<td>0.10</td>
<td>1.80</td>
<td>0.33</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>6.90</td>
<td>0.93</td>
<td>0.93</td>
<td>0.07</td>
<td>1.75</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>7–9</td>
<td>1</td>
<td>8.00</td>
<td>0.90</td>
<td>0.70</td>
<td>0.30</td>
<td>1.80</td>
<td>0.32</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>8.00</td>
<td>0.90</td>
<td>0.90</td>
<td>0.10</td>
<td>1.70</td>
<td>0.28</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>8–10</td>
<td>1</td>
<td>8.90</td>
<td>0.97</td>
<td>1.00</td>
<td>0.00</td>
<td>1.80</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>8.90</td>
<td>0.97</td>
<td>0.97</td>
<td>0.03</td>
<td>1.83</td>
<td>0.30</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Notes: All simulations were run 1 and 1,000 times. The first row represents the hand-crafted adaptation–capabilities mapping. Capabilities per adaptation (CPA) is a measure of how connected the mapping/graph was. “Range” is the size of capability sample selected for each adaptation when constructing this mapping (carried out once per simulation run).
Table 8.4: Results of mapping simulations in discovery mode: are appropriate adaptations suggested?

<table>
<thead>
<tr>
<th>Range</th>
<th>Runs</th>
<th>Caps. per adaptation</th>
<th>Capability coverage</th>
<th>Included</th>
<th>Excluded</th>
<th>Spurious</th>
<th>Top-half</th>
<th>Included</th>
<th>Excluded</th>
<th>Spurious</th>
<th>Top-half</th>
<th>Discovery success</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>1</td>
<td>2.71</td>
<td>0.62</td>
<td>0.50</td>
<td>0.50</td>
<td>0.20</td>
<td>1.00</td>
<td>0.00</td>
<td>0.71</td>
<td>0.08</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>2.71</td>
<td>0.62</td>
<td>0.55</td>
<td>0.45</td>
<td>0.82</td>
<td>0.25</td>
<td>1.00</td>
<td>0.00</td>
<td>0.96</td>
<td>0.11</td>
<td>0.89</td>
</tr>
<tr>
<td>0–2</td>
<td>1</td>
<td>1.30</td>
<td>0.31</td>
<td>0.30</td>
<td>0.70</td>
<td>0.20</td>
<td>0.20</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
<td>0.17</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>1.30</td>
<td>0.31</td>
<td>0.29</td>
<td>0.71</td>
<td>0.15</td>
<td>0.07</td>
<td>1.00</td>
<td>0.00</td>
<td>0.42</td>
<td>0.05</td>
<td>0.80</td>
</tr>
<tr>
<td>1–3</td>
<td>1</td>
<td>1.90</td>
<td>0.38</td>
<td>0.30</td>
<td>0.70</td>
<td>0.30</td>
<td>0.17</td>
<td>1.00</td>
<td>0.00</td>
<td>0.67</td>
<td>0.20</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>1.90</td>
<td>0.38</td>
<td>0.34</td>
<td>0.66</td>
<td>0.33</td>
<td>0.20</td>
<td>1.00</td>
<td>0.00</td>
<td>0.73</td>
<td>0.11</td>
<td>0.86</td>
</tr>
<tr>
<td>2–4</td>
<td>1</td>
<td>2.80</td>
<td>0.62</td>
<td>0.80</td>
<td>0.20</td>
<td>0.60</td>
<td>0.14</td>
<td>1.00</td>
<td>0.00</td>
<td>0.67</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>2.80</td>
<td>0.62</td>
<td>0.55</td>
<td>0.45</td>
<td>0.69</td>
<td>0.24</td>
<td>1.00</td>
<td>0.00</td>
<td>0.65</td>
<td>0.05</td>
<td>0.84</td>
</tr>
<tr>
<td>3–5</td>
<td>1</td>
<td>4.10</td>
<td>0.76</td>
<td>0.80</td>
<td>0.20</td>
<td>1.30</td>
<td>0.19</td>
<td>1.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>4.10</td>
<td>0.76</td>
<td>0.69</td>
<td>0.31</td>
<td>1.10</td>
<td>0.26</td>
<td>1.00</td>
<td>0.00</td>
<td>0.55</td>
<td>0.05</td>
<td>0.98</td>
</tr>
<tr>
<td>4–6</td>
<td>1</td>
<td>5.20</td>
<td>0.83</td>
<td>0.90</td>
<td>0.10</td>
<td>1.40</td>
<td>0.17</td>
<td>1.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.10</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>5.20</td>
<td>0.83</td>
<td>0.76</td>
<td>0.24</td>
<td>1.37</td>
<td>0.28</td>
<td>1.00</td>
<td>0.00</td>
<td>0.43</td>
<td>0.04</td>
<td>1.00</td>
</tr>
<tr>
<td>5–7</td>
<td>1</td>
<td>5.60</td>
<td>0.86</td>
<td>0.70</td>
<td>0.30</td>
<td>1.70</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.10</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>5.60</td>
<td>0.86</td>
<td>0.79</td>
<td>0.21</td>
<td>1.44</td>
<td>0.28</td>
<td>1.00</td>
<td>0.00</td>
<td>0.39</td>
<td>0.05</td>
<td>1.00</td>
</tr>
<tr>
<td>6–8</td>
<td>1</td>
<td>6.80</td>
<td>0.97</td>
<td>1.00</td>
<td>0.00</td>
<td>1.40</td>
<td>0.25</td>
<td>1.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>6.80</td>
<td>0.97</td>
<td>0.90</td>
<td>0.10</td>
<td>1.68</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.32</td>
<td>0.02</td>
<td>1.00</td>
</tr>
<tr>
<td>7–9</td>
<td>1</td>
<td>8.30</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>1.90</td>
<td>0.31</td>
<td>1.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>8.30</td>
<td>1.00</td>
<td>0.96</td>
<td>0.04</td>
<td>1.80</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.28</td>
<td>0.02</td>
<td>1.00</td>
</tr>
<tr>
<td>8–10</td>
<td>1</td>
<td>9.10</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>1.90</td>
<td>0.28</td>
<td>1.00</td>
<td>0.00</td>
<td>0.25</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>9.10</td>
<td>1.00</td>
<td>0.97</td>
<td>0.03</td>
<td>1.84</td>
<td>0.30</td>
<td>1.00</td>
<td>0.00</td>
<td>0.27</td>
<td>0.03</td>
<td>1.00</td>
</tr>
<tr>
<td>1–5</td>
<td>1</td>
<td>3.40</td>
<td>0.76</td>
<td>0.80</td>
<td>0.20</td>
<td>0.90</td>
<td>0.24</td>
<td>1.00</td>
<td>0.00</td>
<td>0.67</td>
<td>0.10</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>3.40</td>
<td>0.76</td>
<td>0.67</td>
<td>0.33</td>
<td>1.03</td>
<td>0.26</td>
<td>1.00</td>
<td>0.00</td>
<td>0.64</td>
<td>0.07</td>
<td>0.84</td>
</tr>
<tr>
<td>1–10</td>
<td>1</td>
<td>6.00</td>
<td>0.86</td>
<td>0.80</td>
<td>0.20</td>
<td>1.60</td>
<td>0.25</td>
<td>1.00</td>
<td>0.00</td>
<td>0.67</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td>6.00</td>
<td>0.86</td>
<td>0.82</td>
<td>0.18</td>
<td>1.52</td>
<td>0.29</td>
<td>1.00</td>
<td>0.00</td>
<td>0.49</td>
<td>0.06</td>
<td>1.00</td>
</tr>
</tbody>
</table>

Notes: All simulations were run 1 and 1,000 times. The first row represents the hand-crafted adaptation-capabilities mapping. Capabilities per adaptation (CPA) is a measure of how connected the mapping/graph was. “Range” is the size of capability sample selected for each adaptation when constructing this mapping (carried out once per simulation run). “Discovery success” indicates if the randomly-removed adaptations from the usage log are suggested.
Algorithm 8.1 create\_mapping\((n, r_0, r_1)\).

**Require:** Number \(n\) of adaptations to create.
Range denoted by \(r_0\) and \(r_1\) of capabilities per adaptation.
Global set of capabilities \(C\).

**Ensure:** A mapping from adaptations to capabilities is returned with the desired capabilities per adaptation value.

1: \(\text{map} \leftarrow \emptyset\)
2: **for** \(i = 0\) to \(n\) **do**
3: \hspace{1em} Sample a set \(K\) of random capabilities from \(C\) where \(r_0 \leq |k| \leq r_1\).
4: \hspace{1em} Create a random name for this adaptation.
5: \hspace{1em} Add \((\text{name}, K)\) to \(\text{map}\).
6: **end for**
7: **return** \(\text{map}\)

---

8.4.3 Extent of Adaptations

The trivial mapping technique introduced in section 6.9 does not reflect the real-world scenario that the *extent* of an adaptation has implications as to the level of capability implied and, vice-versa, that different magnitudes of capability fluctuation may require adaptations of different “strengths” (also: there is not necessarily a linear relationship between capability fluctuation and appropriate strength of adaptation, for a given person, at a given time, in a given situation).

Figure 6.14 introduced the mapping between adaptations and capabilities. This is extended by Figure 8.5 to model different “levels” of adaptations and their implications to certain capabilities. For example: screen magnification and/or minimum font sizes of different levels (indirectly) imply differing levels of visual acuity.

This extended model is actually equivalent to the original one, as the mul-
tiple, linked, levels of an adaptation or capability may be “flattened” into distinct adaptations or capabilities, thus preserving the qualities of the original model.

8.4.4 Specificity of Capabilities

The model of extent, above may be used to signify that different “levels” of an adaptation can have different magnitudes of effect. It would be more useful if these magnitudes could be mapped onto real-world human capability values—or, in the least, ranges. As adaptations affect the system in technical ways, a process is needed to convert technical measurements into the relevant units of human capability. This is a common activity in the CAAR system so it is provided by it in the form of an “oracle” process.\(^3\)

Further information may be required to map technical changes to human capability values (or, more likely, ranges of capability values). Such mapping would be carried out by the user model in the relevant adaptations.\(^4\) Figure 8.6 presents the full range of situations in which adaptations are mapped to capabilities, examples of which follow.

Trivial: colour deficit filter indicates problems with colour sensitivity capabilities.

Levels: (Due to the need to consider the effects of other concurrent adaptations, this notional model can only be provided by the use of a mediating process, as follows, because if adaptations have access to change system parameters, these may already have been changed by other adaptations.)

Oracle: decreasing font size indicates greater visual acuity than no change to font size, or an increase.

To enable comparison of the level of adaptation, the parameter of font size, following all adaptations in the system, must be known. As it can be affected by all adaptations in the system (including font-size changes and full-screen magnification), the oracle must be consulted to determine the font size—which is related closely to visual acuity, but it is not possible to ascertain visual acuity directly from only this (technical) measurement.

The result, e.g. font size, would only be applicable on a particular T-device at best.

Oracle and Model: the oracle (run-time technical component) can supply information on the point-size of a particular piece of text, given the effects of

\(^3\)In a production system, the oracle would be part of the run-time support system running on the local machine, as described in Appendix B—this supports the separation between reasoning process and computing platform.

\(^4\)The proposed run-time structure of adaptations is discussed in Appendix B.
Adaptation Capability

Adaptation level 1

Adaptation level 2

Technical measurement (Oracle)

Human measurement (Model)

Capability level 1

Capability level 2

Capability range 1

Capability range 2

Figure 8.6: Four adaptation–capability mapping types: direct; magnitudes of adaptations implicating different capabilities; mapping via an oracle process which acts as a gatekeeper to the technical quantity being manipulated by the adaptation (e.g. OS base widget font size); mapping via an oracle and a model that converts the technical quantity into real-world or human terms (e.g. font size and estimated viewing distance to visual acuity).
all engaged adaptations. Further, the resolution and physical size\textsuperscript{5} of the display device in use can be ascertained.

This information would be passed to the model, which would estimate a feasible range for the user’s visual acuity, based on likely distance from monitor and any other known relevant capabilities.

Alternatively, a simple mapping, based on the technical information obtained from the oracle, to the physical font size (i.e. in centimetres) could be performed.

The results from either involved or trivial model, are portable because they are in human-capability or real-world units. Therefore they are useful for when the user wishes to move to a new T-device, as appropriate adaptations may be applied at render-time to enable the user to perceive information transmitted via the new T-device.

The benefits of being able to ascribe likely ranges to capability values are clear—adaptation suggestions can be more refined. As with the trivial mappings introduced previously, the principle of “triangulation” over time also applies: capability ranges may be predicted using multiple oracles and adaptations’ user models, on multiple devices. This allows the overall prediction to be refined over time in the same way as described above for general capability implications.

In terms of algorithmic complexity, this process is still relatively trivial (it is simply a mapping with two, as opposed to one, look-up stages). In practical terms, it will clearly require more execution time; however, this too is relatively trivial and may be mitigated by the following.

- The OS must track the base value for parameters such as: screen brightness; system/application sound output volume; sizes and colours of rendered elements—and adaptations that modify these often do so in a very predictable way.

- The results from calls to oracles may be cached. When an additional related adaptation is invoked or disengaged, the value of the relevant parameter (e.g. font size) can be updated.

- The results in terms of human capability ranges may be stored as literals, rather than re-calculated when analysing the data in future. Whilst each individual predicted capability range may not be completely accurate, the cumulative effect of many predictions will allow estimates to tend to the correct value, as with the trivial mapping, above.

\textsuperscript{5}Not necessarily always known at present, but possible for manufacturers to supply.
In order for adaptations to be able to expose their respective user models, and express their effects in technical terms, the following components are mandated.

**Technical effects:** E.g. the use of full-screen magnification at a linear scale factor of two implies that all font point-sizes will be twice as large as before the adaptation was invoked.

Standard technical units, such as those given in [64] would be used. The effects on these may be computed by using a technical model in the adaptation, or simple modifiers such as scaling (as would be the case with full-screen magnification).

**Model:** used to predict human capability ranges, based on available technical information.

The standard capability classification (ICF, possibly extended) is used.

An “**Affector**” to perform the adaptation; this may be the only platform-specific part of the adaptation.

Further, the run-time process(es) which embody the oracle in a production system would need to maintain a “blackboard” architecture, in order to allow information from multiple engaged adaptations to be usable at any time—for more details, see subsection B.2.2.

### 8.4.5 Cross-device Portability

The suitability of adopting a capability-based approach is demonstrated using the example of font sizing. The first stop to moving to capability-based reasoning is to use real-world units of measure instead of device-specific ones; instead of expressing font sizes in points, they can be expressed in standard measures of length.

The actual Dots Per Inch (DPI) of a device can be trivially calculated using Pythagoras’ theorem if its pixel resolution and diagonal measure in real-world terms are known. As many devices display fonts based on their “point” as opposed to pixel sizes, where points are \( \frac{1}{72} \) of an inch, usually scaled as if the display were operating at 96 DPI.

As can be seen from Table 8.5, it is possible to ensure a consistent real-world font size that is suitable for the user, based on the size of the device and its resolution. However, a sensible minimum font size depends on the distance between user and device—clearly 1cm-high text on a cinema screen probably will not be legible to a person who requires it on a smartphone. This is where the utility
Table 8.5: Examples of cross-device font sizing.

<table>
<thead>
<tr>
<th>Name</th>
<th>Diagonal (inches)</th>
<th>Size (pixels)</th>
<th>DPI</th>
<th>14-point text (inches)</th>
<th>1cm text (points)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cinema (4K, 2.39:1)</td>
<td>300</td>
<td>4096 × 1716</td>
<td>15</td>
<td>1.26</td>
<td>3.20</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>50</td>
<td>1280 × 720</td>
<td>29</td>
<td>0.64</td>
<td>1.61</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>42</td>
<td>1280 × 720</td>
<td>35</td>
<td>0.53</td>
<td>1.36</td>
</tr>
<tr>
<td>1080p HD Television</td>
<td>50</td>
<td>1920 × 1080</td>
<td>44</td>
<td>0.42</td>
<td>1.08</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>32</td>
<td>1280 × 720</td>
<td>46</td>
<td>0.41</td>
<td>1.03</td>
</tr>
<tr>
<td>1080p HD Television</td>
<td>42</td>
<td>1920 × 1080</td>
<td>52</td>
<td>0.36</td>
<td>0.90</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>42</td>
<td>1280 × 720</td>
<td>52</td>
<td>0.36</td>
<td>0.90</td>
</tr>
<tr>
<td>1080p HD Television</td>
<td>50</td>
<td>1920 × 1080</td>
<td>69</td>
<td>0.27</td>
<td>0.69</td>
</tr>
<tr>
<td>iMac</td>
<td>24</td>
<td>1600 × 1000</td>
<td>79</td>
<td>0.24</td>
<td>0.60</td>
</tr>
<tr>
<td>iMac</td>
<td>24</td>
<td>1920 × 1200</td>
<td>94</td>
<td>0.20</td>
<td>0.50</td>
</tr>
<tr>
<td>iPhone</td>
<td>3.5</td>
<td>320 × 480</td>
<td>165</td>
<td>0.11</td>
<td>0.29</td>
</tr>
<tr>
<td>iPhone 4</td>
<td>3.5</td>
<td>640 × 960</td>
<td>330</td>
<td>0.06</td>
<td>0.14</td>
</tr>
</tbody>
</table>

Table 8.6: Example T-device viewing distances.

<table>
<thead>
<tr>
<th>Name</th>
<th>Diagonal (inches)</th>
<th>Minimum (m)</th>
<th>Maximum (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cinema (4K, 2.39:1)</td>
<td>300</td>
<td>10.000</td>
<td>20.000</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>50</td>
<td>2.000</td>
<td>3.000</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>42</td>
<td>2.000</td>
<td>3.000</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>32</td>
<td>1.400</td>
<td>2.000</td>
</tr>
<tr>
<td>iMac</td>
<td>24</td>
<td>0.400</td>
<td>0.600</td>
</tr>
<tr>
<td>iPhone</td>
<td>3.5</td>
<td>0.150</td>
<td>0.400</td>
</tr>
</tbody>
</table>

An estimate of visual acuity can be continuously updated based on the known font-size settings and likely distance between the user and device (indicated by the device class). Table 8.6 gives likely viewing distance ranges. Given usage logs such as the the following, containing records in the form (device, minimum font size), it is easy to compute and continuously update an average “font height in metres per metre viewing distance” metric, which closely models that of visual acuity.

(iMac, 0.006)

(iPhone, 0.004)

(720p HD Television, 0.020)

Examples based on this log fragment are given in Table 8.7. This metric can
Table 8.7: Accuities related to T-devices.

<table>
<thead>
<tr>
<th>Name</th>
<th>Average view distance (m)</th>
<th>Font size (m)</th>
<th>Acuity</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMac</td>
<td>0.500</td>
<td>0.006</td>
<td>0.012</td>
</tr>
<tr>
<td>iPhone</td>
<td>0.275</td>
<td>0.004</td>
<td>0.015</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>1.700</td>
<td>0.020</td>
<td>0.012</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>0.013</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 8.8: Predicted font sizes with acuity 0.013.

<table>
<thead>
<tr>
<th>Name</th>
<th>Ideal Size (cm)</th>
<th>Predicted (cm)</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>iMac</td>
<td>0.60</td>
<td>0.64</td>
<td>1.06</td>
</tr>
<tr>
<td>iPhone</td>
<td>0.40</td>
<td>0.35</td>
<td>0.88</td>
</tr>
<tr>
<td>720p HD Television</td>
<td>2.00</td>
<td>2.17</td>
<td>1.09</td>
</tr>
</tbody>
</table>

then be used to apply a minimum font size setting to a new device, even if it is of a form-factor the user has not yet encountered. Given the logs above, the acuity-like metric was calculated for each device, based on the average viewing distance. It was then applied to the above devices, yielding the results shown in Table 8.8—these were not totally accurate, but certainly meet the goal of bootstrapping a user onto a new device.

Medical visual acuity is calculated in a similar way, albeit with highly specified stimulus data in the form of particular glyph shapes and a prescribed distance from the stimulus. However the goal in this case is to arrive as unobtrusively as possible at a solution at least “good enough” to allow a new device to be bootstrapped, which can benefit from known more precise values if they are supplied.

8.4.6 Relation to Collaborative Situations

It is assumed that users wishing to collaborate will already have established adaptation preferences that may be applied to the collaboration (i.e. users will already have favourite adaptations appropriate for the contexts involved in the collaboration). Because of this, the reasoning regarding compatibility in collaborations is carried out based upon these preferences.

Should it be desired, it would be possible to combine the outputs of the mapping process, for distinct users, in the following ways.

- Filtering capabilities to ascertain which are in common between users, or which are possessed only be one user. (This is analogous to, and would be computed as, basic set operations.)
• Filtering adaptation suggestions in order to suggest adaptations that may be suitable to both users. (For the reasons above, this is not the approach taken to collaborative scenarios—unless no compatible adaptations can be found within users’ existing preferred adaptation lists.)

8.5 Usage-based Preference Deduction and Compatibility

Tests were conducted to assess how to ascertain users’ TRIALS priorities from usage logs of the abstract adaptations underlying the concrete adaptations actually invoked. Several sample usage logs were constructed by selecting sets of valid abstract adaptations to fix a lack of capacity in one channel. The overall TRIALS indicators for each set of abstract adaptations was noted. Usage logs were of the following form.

\[
\text{log} := ((\text{adaptation set, time engaged}), \ldots)
\]

Algorithm 8.2 indicator_scores(usage log).

Require: Usage log allowing look-up of the usage count and TRIALS indicators for a given adaptation set.
The total usage count of all adaptation sets in the log.
Ensure: A mapping, from each value that each indicator can take to its score, is returned. The score indicates how favoured that value is for that indicator by the user.

1: map has a storage slot for each (indicator, value) combination.
2: for each TRIALS indicator do
3:     for each possible value of the indicator do
4:         for each adaptation set in the log do
5:             if current indicator has current value in the TRIALS consequences of this set of adaptations then
6:                 Add this adaptation set’s usage count to map(indicator, value).
7:             end if
8:         end for
9:     Normalise the total score for this indicator and value by dividing by the usage count of all adaptations.
10: end for
11: end for
12: return map

Based on these usage logs, a score for each indicator was computed to effectively give the likelihood that it would be at the top of the user’s usage distribution, using
algorithm 8.2. Each sample usage log was then ranked according to its score, rather than actual usage count. Table 8.9 shows some typical results. The “error” is the number of places out-of-rank each prediction is compared to the actual rank by usage count. The scores for each factor are given below.

While simple, the approach is effective. However this is under the assumption of a relatively non-linear usage profile; it is assumed that most users will settle on a suitable set of (underlying abstract) adaptations in a given context, thus the distribution will be skewed to promote a small number of sets of adaptations above any others. In most cases, this assumption seems reasonable. If, however, the usage profile is more linear—as a result of exploring available adaptation suggestions, for example—this technique does not work as well, as shown in Table 8.10. There are two potential solutions to this.

- Use a more robust classification technique such as a feature selector or perhaps ID3, an entropy-based decision tree generator. This would have the effect of ascertaining which are the most important indicator values.

- Use the present indicator scores in conjunction with other data, such as aggregated adaptation suggestions from other users on the same device or in the same context.

The course of action to take would depend on the shapes of the adaptation set distributions that naturally occur in real-world use.
Table 8.9: Results of abstract adaptation usage analysis (ranking error and indicator scoring factors).

<table>
<thead>
<tr>
<th>Score</th>
<th>Usage Rank</th>
<th>Error</th>
<th>Usage Count</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
<th>Adapts</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.800</td>
<td>1</td>
<td>0</td>
<td>50</td>
<td>↑</td>
<td>-</td>
<td>-</td>
<td>○</td>
<td>●</td>
<td>↑</td>
<td>Share (V), Reflow (V), Pan (V)</td>
</tr>
<tr>
<td>4.440</td>
<td>2</td>
<td>0</td>
<td>40</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>○</td>
<td>●</td>
<td>↑</td>
<td>Supplement (A), Share (V), Reflow (V), Pan (V)</td>
</tr>
<tr>
<td>4.360</td>
<td>3</td>
<td>0</td>
<td>20</td>
<td>↑</td>
<td>-</td>
<td>-</td>
<td>○</td>
<td>○</td>
<td>↑</td>
<td>Share (V), Pan (V)</td>
</tr>
<tr>
<td>4.000</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>○</td>
<td>○</td>
<td>↑</td>
<td>Supplement (A), Share (V), Pan (V)</td>
</tr>
<tr>
<td>2.000</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>○</td>
<td>○</td>
<td>-</td>
<td>Replace (A)</td>
</tr>
<tr>
<td>0.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑ or ●</td>
<td>0.96</td>
<td>0.40</td>
<td>0.40</td>
<td>0.00</td>
<td>0.72</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>0.04</td>
<td>0.60</td>
<td>0.56</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>↓ or ○</td>
<td>0.00</td>
<td>0.00</td>
<td>0.04</td>
<td>1.00</td>
<td>0.28</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Symbols: ● = yes; ○ = no; - = not applicable
Table 8.10: The simple indicator-scoring algorithm does not accommodate more linear usage profiles.

<table>
<thead>
<tr>
<th>Score</th>
<th>Usage Rank</th>
<th>Error</th>
<th>Usage Count</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
<th>Adaptations</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.645</td>
<td>1</td>
<td>0</td>
<td>1000</td>
<td>↑</td>
<td>–</td>
<td>–</td>
<td>○</td>
<td>•</td>
<td>↑</td>
<td>Share (V), Reflow (V), Pan (V)</td>
</tr>
<tr>
<td>4.484</td>
<td>3</td>
<td>1</td>
<td>700</td>
<td>↑</td>
<td>–</td>
<td>–</td>
<td>○</td>
<td>○</td>
<td>↑</td>
<td>Share (V), Pan (V)</td>
</tr>
<tr>
<td>4.355</td>
<td>2</td>
<td>-1</td>
<td>800</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>○</td>
<td>•</td>
<td>↑</td>
<td>Supplement (A), Share (V), Reflow (V), Pan (V)</td>
</tr>
<tr>
<td>4.194</td>
<td>4</td>
<td>0</td>
<td>500</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>○</td>
<td>○</td>
<td>↑</td>
<td>Supplement (A), Share (V), Pan (V)</td>
</tr>
<tr>
<td>2.097</td>
<td>5</td>
<td>0</td>
<td>100</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>○</td>
<td>•</td>
<td>↑</td>
<td>Replace (A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Score</th>
<th>Usage Rank</th>
<th>Error</th>
<th>Usage Count</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
<th>Adaptations</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.275</td>
<td>3</td>
<td>2</td>
<td>800</td>
<td>↑</td>
<td>–</td>
<td>–</td>
<td>○</td>
<td>○</td>
<td>↑</td>
<td>Share (V), Pan (V)</td>
</tr>
<tr>
<td>4.225</td>
<td>1</td>
<td>-1</td>
<td>1000</td>
<td>↑</td>
<td>–</td>
<td>–</td>
<td>○</td>
<td>•</td>
<td>↑</td>
<td>Share (V), Reflow (V), Pan (V)</td>
</tr>
<tr>
<td>4.025</td>
<td>4</td>
<td>1</td>
<td>700</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>○</td>
<td>○</td>
<td>↑</td>
<td>Supplement (A), Share (V), Pan (V)</td>
</tr>
<tr>
<td>3.975</td>
<td>2</td>
<td>-2</td>
<td>900</td>
<td>↑</td>
<td>↑</td>
<td>↑</td>
<td>○</td>
<td>•</td>
<td>↑</td>
<td>Supplement (A), Share (V), Reflow (V), Pan (V)</td>
</tr>
<tr>
<td>2.575</td>
<td>5</td>
<td>0</td>
<td>600</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>○</td>
<td>•</td>
<td>↑</td>
<td>Replace (A)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.00</td>
</tr>
</tbody>
</table>

Symbols: • = yes; ○ = no; – = not applicable
8.5.1 Compatibility Between Users

Figure 8.7 illustrates a scenario in which two users have differing abstract adaptation usage profiles. These would each be analysed as in the previous section to give indicator scores. The “spread” between scores for each user is calculated as follows.

\[ \text{spread} = \frac{\max (\text{scores}) - \min (\text{scores})}{\sum \text{scores}} \] (8.1)

where \(\text{scores}\) is the set of users’ scores.

Several different adaptation sets were costed for users with differing indicator scores. Table 8.11 shows one example of the results of comparing TRIALS consequences between users. The spread metric attempts to show that, despite apparently similar costs, there are substantial differences between users’ opinions on some adaptation sets.

8.6 Compatibility with Existing Work

The work carried out here should ideally be adoptable as well as informative for other researchers and developers. Whilst Sus-IT is carrying out future development, it is important to discuss how the design of the system developed in this thesis is compatible with other work.

8.6.1 Abstract Interface Generation

Gajos’ SUPPLE and similar systems: CAAR never went into the detailed widget mapping that Gajos’ work performed, as this was not necessary; the goal here was to show that an approach could be developed that wrapped around and could be integrated with the likes of SUPPLE. Much of the reasoning of CAAR is both simple, tractable and expresses the problems in similar ways to SUPPLE, so it could indeed be thought of as an extension of that work.

8.6.2 Adaptations

The interface—both programmatic and metadata—between the reasoning process (and OS) and adaptations is intentionally very simple (subsection 6.10.3). It does not supplant any existing assistive APIs. It only requires capability metadata to be associated with adaptations, applications and devices. Whilst this may appear to be considerable extra work, the data may be specified to varying degrees, as
Table 8.11: User indicator scores and score prediction.

<table>
<thead>
<tr>
<th>Value</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑ or ●</td>
<td>0.96</td>
<td>0.40</td>
<td>0.40</td>
<td>0.00</td>
<td>0.72</td>
<td>0.96</td>
</tr>
<tr>
<td>-</td>
<td>0.04</td>
<td>0.60</td>
<td>0.56</td>
<td>-</td>
<td>-</td>
<td>0.04</td>
</tr>
<tr>
<td>↓ or ○</td>
<td>0.00</td>
<td>0.00</td>
<td>0.04</td>
<td>1.00</td>
<td>0.28</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adaptation Set</th>
<th>User 1 Score</th>
<th>User 2 Score</th>
<th>Spread</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share (V), Pan (V)</td>
<td>4.360</td>
<td>4.879</td>
<td>0.06</td>
</tr>
<tr>
<td>Supplement (A), Share (V), Pan (V)</td>
<td>4.000</td>
<td>4.061</td>
<td>0.01</td>
</tr>
<tr>
<td>Share (V), Reflow (V), Pan (V)</td>
<td>4.800</td>
<td>3.879</td>
<td>0.11</td>
</tr>
<tr>
<td>Supplement (A), Share (V), Reflow (V), Pan (V)</td>
<td>4.440</td>
<td>3.061</td>
<td>0.18</td>
</tr>
<tr>
<td>Reduce (V)</td>
<td>2.000</td>
<td>3.394</td>
<td>0.26</td>
</tr>
<tr>
<td>Replace (A)</td>
<td>2.000</td>
<td>3.394</td>
<td>0.26</td>
</tr>
<tr>
<td>Supplement (A)</td>
<td>2.160</td>
<td>2.909</td>
<td>0.15</td>
</tr>
</tbody>
</table>
ICF for example is hierarchical. There are many community projects that aim to help users of devices by cataloguing their capabilities;\footnote{http://wurfl.sourceforge.net/} capabilities could be supported in a similar way for legacy products.

### 8.6.3 Standards and Projects

**WHO ICF.** Compatible as it takes the same capability approach.

**ISO 24571.** Compatible as the standard defines technical changes that can be made by adaptations, of which the reasoner must be aware.

**WAI WCAG** standards are complementary to capability-based reasoning. Whilst the fundamentals—e.g. “POUR” (see section 2.5)—are exemplary, there is a bias towards certain disabilities and lack of flexibility. It is hoped that a more minimal WAI WCAG, supplemented with capability-based reasoning, could go a long way towards solving these problems (more user-focused).

**SNAPI.** The standard for encoding user preferences is now very widespread. In the UK, it is present on all bus passes, as they use the “ITSO” smartcard standard. The SNAPI encoding is presently very device-specific, but it can be extended and does have an extensive user-base already. It could perhaps be used as a “staging post” in delivering capability-based reasoning.

**GPII.** As introduced in section 2.13, this project is developing means for ATs (in fact Vanderheiden’s “micro-ATs”, which are similar to adaptations) to be delivered universally in part via web services. There is a focus on adapting systems according to users’ preferences, which include modality preferences and technical settings. This makes the GPII an ideal delivery mechanism for CAAR, which would provide capability-based bootstrapping across devices and user awareness-raising regarding available adaptations which, as discussed, would fit the remit of the “match maker” component of the delivery infrastructure.

### 8.7 Summary

This chapter has presented technical justifications and mechanical analysis of key aspects of the capability-based reasoning process. The results obtained demonstrate the efficacy of the process and suggest some potential avenues for further development towards a “production-ready” implementation.
Chapter 9

Participant-based Focussed and Longitudinal Test Designs

The focus of this thesis has been to develop a technical approach to reasoning about capabilities and adaptations. This approach has been mechanically assessed for feasibility and some exploration as to how to extend the reasoning temporally has been carried out.

However, as the process developed is designed to benefit people, the system will be assessed for acceptance with real participants by the Sus-IT project. Whilst it would not be practicable to carry out the required extensive tests as part of this thesis, it is important to demonstrate the work that is ongoing as part of the wider project.

This chapter describes a number of avenues of investigation that could be carried out to assess the real-world efficacy of the system, some of which are being explored by Sus-IT. As the previous chapter sought to demonstrate the technical potential of the system to fulfill the requirements, the tests described here seek to show, over a period of time, that the system can work in the real world.

9.1 Overview

Whilst most of the tests in the previous chapter were objective in nature, there are both subjective and objective assessments that can help answer these questions. Objective tests include the following.

- Reasoning accuracy at an instant (i.e. matching people to adaptations in different known circumstances).
- Capability tracking accuracy (reactions to DUET constraints, adaptations affected by users and user preferences).
• Longitudinal usage (assessing the effectiveness of the system over time for given users; collecting capability data on the users).

Appropriate subjective tests are as follows.

• Reactions to the concept (already carried out, through interactive theatre [101] and sandpit device creation sessions).

• Exploratory workflow study—assessing and refining the UI and mechanics of interaction.

• Collaboration (experience of people of different abilities working together).

• Longitudinal usage (qualitative—users’ experiences over time).

• Users’ experience of the stability reasoning aspects.

Part of the longer-term strategy for Sus-IT, to be carried out jointly by all work-packages, includes the assessment of the system with developers of ATs and mainstream software, as well as content authors and providers. Designs for some key tests are given later in this chapter.

9.1.1 General Challenges and Approach

Whilst this work is not aimed exclusively at people with recognised disabilities, there is a strong intention to support such people. There are often significant logistical constraints on organising participator tests involving people with recognised disabilities. In fact, it could be even more difficult to find a sample of participants that are expected to experience minor-to-moderate impairments naturally at certain points in the assessment of the system. One of the drawbacks of being unable to directly measure capability fluctuations is that in order to test the reasoning process, these fluctuations must be expected.

Therefore it seems clear that some tests—those of the most fundamental reasoning processes regarding reaction to and tracking of capabilities as well as mapping them to adaptations—would be better performed under considerably more artificial conditions, in which case the sample of participants selected would ideally be expected to undergo as few capability fluctuations as possible during the assessments of the reasoning process, so that the process is the only variable under test.

A further, significant, challenge is the truly vast, if not infinite, scale of possible scenarios in which the interplay between adaptations may present itself. As stated in the previous chapter, the goal is not to test the adaptations themselves,
rather the interference between them. In fact, as this is variable depending on the perceptions of the individuals involved, a more reasonable goal would be to ascertain reasonable limits on the variation in the perceptions of that interference.

### 9.1.2 Participant Groups

A range of different participant groups may participate in testing the system, as follows.

**“Non-PC users”** as in previous Sus-IT testing this group was found to have different attitudes to adaptations than expert computer users [101]. People in this group also have fewer expectations brought about by experience and bring less of an existing mental model of how technology works—presenting the opportunity to assess adaptivity “cleanly”.

**Seasoned computer users** may present an ideal opportunity to glean feedback from someone whom can articulate opinions on technical matters more comprehensively. However, there is an inherent and substantial challenge in overcoming established expectations and behaviours—in the case of adaptivity, particularly if tests are being carried out on legacy devices with which the user is familiar, this can create barriers to engagement as the system may be, perhaps subconsciously, perceived as unnecessary.

**People with recognised disabilities.** Whilst this work is not aimed exclusively at supporting people with disabilities, some useful insights may be gleaned—and perhaps more rapidly—by someone experiencing known moderate and/or fluctuating accessibility barriers.

**People without recognised disabilities.** People of all ages, without recognised disabilities, but possibly with minor-to-moderate impairments, are the main focus of this work. For this reason, recruitment of participants can be from the general public or, as is often popular due to resource constraints typical of proof-of-concept testing, coupled with the need for competent users, population of the university. However, there are two caveats in this case.

- When recruiting people from a constrained population, such as university staff or postgraduate students, it must be borne in mind that whilst a reasonable cross-section of sensory and motor capabilities may be obtained, the backgrounds of the people sampled will likely be very similar in terms of education and experience of technology. This makes such a group alone unsuitable for testing matters of interaction workflow, because the system is designed for use by the wider population.
• When testing particularly the fundamental reasoning elements of the system, people who are less likely to experience capability fluctuations are more desirable participants. This is because fewer unknown variables are involved. Such participants may be of great help in calibrating the system and taking part in scenarios that simulate impairment or capability fluctuation.

9.1.3 Technical Challenges

A number of technical challenges in implementing a test CAAR system and corresponding UI and adaptations present themselves.

Most applications are not abstract. It is not currently possible to apply adaptations, such as fold, to most applications’ interfaces (reductions may be applicable to very scriptable applications such as Microsoft Word or Mozilla Firefox). There are no known totally abstract applications (i.e. those in which every entity may be accessed and mutated), nor any mainstream toolkits for creating multi-modal abstract adaptations.

This problem is overcome in two ways: (1) concentrating on the most popular and mutable applications in the first instance (web browsers), when testing and demonstrating the benefits of adaptivity reasoning and (2) taking advantage of accessibility features within the host OS that may be accessed programmatically (such as variable font size for widgets, and TTS).

ATs are not abstract. As they are designed for static, chronic impairments, ATs can be invasive and take over an entire system.

This can be partially overcome by targeting more mutable applications and by creating a minimal number of ATs from more basic components, such as TTS. Code for existing, open, ATs could be instrumental in this.

Most ATs are expensive. Whilst this is true, it has been possible to create a small number of basic ATs in-house, partly by learning from existing open-source projects.

Research ATs can be obscure. Whilst many exceptions to this rule exist (such as MAAVIS [67] and SUPPLE), these are more general ATs; the highly-focused adaptations for specific disabilities are harder to obtain, as they are often used for research projects and not commercialised, due to a perceived lack of market (Huetility\(^1\) being a notable exception).

\(^1\)http://www.huetility.com/
Whilst it is a long-term goal of Sus-IT to cater for more severe impairments, assessment of a CAAR-based system can be carried out without such specialist ATs—though these are being sought out in parallel to the testing described here.

9.1.4 Ethical Considerations

Testing this work involves obtaining information regarding people’s usage of systems that many would regard as personal, regardless of whether the usage data alone may be used to personally identify an individual. Naturally, sound ethical consideration and planning must underpin any participatory research. This work is no exception and requires that participants place a great deal of trust in those monitoring the assessment of the system, as well as those analysing and the storage of results.

A great deal of ethical challenges present themselves when dealing with the monitoring of people, including the following.

- Ensuring that a balanced and sensitive approach is taken regarding any detected capability declines. On balance it seems most sensible to alert users to detected declines—as people may otherwise be unaware of them—but finding a suitable manner in which to do this is a great challenge.

- The possibility of collecting data for one purpose that could be useful to others in diagnosing potential medical problems.

- The responsibility of deciding to notify other parties should serious problems be detected. This must be balanced with the concern expressed by some users that irresponsible (mainly implied as commercial) entities be given access to the user’s information.

These matters are being explored by the wider Sus-ITs project in parallel to the technical work described in this thesis, through the following avenues of exploration.

Interactive theatre and “sandpit” activities as described in the introduction; these are being used to elicit potential users’ feelings with regard to the perceived benefits and pitfalls of adaptive systems.

The development of an ethics framework at a project-wide level, with consultation from all stakeholders—of course, older people.
Workshops and seminars in the research community and with industrial partners, to raise the matters and develop an approach to identify the key research challenges in the area and develop an agenda to tackle them.

9.2 Interaction Workflow

An assessment of the interaction with a system supporting adaptivity could explore the following dimensions.

- The balance of control: user- to system-initiated and whether the system can suitably adjust or be adjusted to accommodate users’ preferences on this (see subsections 6.2.2).
- How adaptations should be represented.
- Are the feedback options significantly expressive?
- Users’ impressions of the overall workflow cycle.

9.2.1 Background: Proposed Workflow

Part of the workflow given in subsection 3.5.3 involves the participants requesting help if they feel this is necessary, as well as using an always-present UI to provide feedback on currently-engaged adaptations (simply explicit acceptance or rejection). A universal way in which to provide this facility was designed: a “help button”, a mock-up of which is depicted in Figure 9.1. When activated, the user would optionally be able to direct the request for help to a specific application (or, in the case of abstract applications, part thereof). Following this, the system would suggest appropriate adaptations, using an interface such as that depicted in Figure 9.2.

The nature of these tests, particularly in the initial stages, necessitates constructive feedback from users who may benefit from the system but are also technically-motivated enough to explore the workflow. Largely, feedback would be qualitative.

9.2.2 Representation of Adaptations

The MAAVIS project has created an easy-to-use interface to common applications suitable for people with dementia [67] (the remit of the project has been expanded to school pupils with learning difficulties). In user trials carried out with people with dementia, it was determined that participants generally did not prefer icons
CHAPTER 9. PARTICIPANT-BASED TEST DESIGNS

Figure 9.1: “Help button” mock-up and Sus-IT software version (which is always accessible on-screen). (Reproduction of Figure 3.18.)

Figure 9.2: Adaptation set selection dialogue box (design and Sus-IT implementation). Advanced users would be given the option to reconfigure the adaptations proposed in a given set, as is shown in this figure—however these options would be hidden by default so that novice users are not overwhelmed. The manner in which previews of the adaptations are affected would vary, but actually applying the adaptations to the system is argued to be the best approach, where possible. (Reproduction of Figure 3.17.)
that were used to represent different applications or services (such as document-writing or voice chat). Many people preferred text labels on buttons, as they were able and keen to read.

Some exploration should be carried out to determine how proposed combinations of adaptations should be (a) presented to the user and (b) may be modified by the user before application. It is hypothesised that, where possible, a “live preview” of adaptations should be used. However, some people are likely to want further details (for information or so that the configuration of adaptations may be refined); in these cases, how should the alternatives be represented? Alternatives include by textual description and abstract or more realistic images denoting the following.

- The mechanism of the adaptations.
- The areas that the adaptations are designed to address.
- The effects of the adaptations on the user.
- The effects of the adaptation on devices (e.g. replacement or altered bandwidth).
- The effects of the adaptation on the application being used—particularly important in collaborative scenarios (e.g. “you will be able to work together on x (application) parts, but not y parts”.)

It is almost certain that a range of preferences for representing adaptations will exist, so it will be necessary to determine the broad dimensions of those preferences (e.g. abstract or realistic; mechanism- or effects-focussed), then devise a heuristic or method for selecting the most appropriate presentational style based on the user, their capabilities and possibly other preferences.

### 9.2.3 Exploratory Test Scenario

The pilot system whose architecture is depicted in Figure 9.3 will be used. Supported applications and adaptations in this system will include: Mozilla Firefox; Microsoft Word; system-level accessibility settings such as double-click speed and mouse sensitivity; UI for reviewing suggested adaptations and UI for requesting help and offering accept/reject feedback on adaptations.

---


3 This is a simpler version of the full architecture, described in Appendix B and depicted in various diagrams in section B.3.
Users will be informed of the existence of the adaptivity system and asked to use the computer as normal, but initiate a request for help whenever they have any accessibility problems. Participants will then be free to engage in any of a particular set of “approved” tasks (email, web browsing, perhaps limited to particular sites for consistency reasons\textsuperscript{4}). Data collected will include:

- Observations of the user’s interactions with the system (qualitative and likely to only be used in initial pilot studies to check the workflow system is functioning correctly and seek constructive feedback from expert users).
- Capability and adaptation events.
- Reactions to suggested adaptations (of the types discussed in section 6.8). Likely to be trialled after a number of sessions.

It would be good practice to carry out initial trials with more expert users, as they are able to describe their opinions on technical matters more comprehensively. Table 9.1 gives details of two potential tests; the first of which is concerned with gaining feedback on the workflow of the adaptive system and the second compares this with the traditional approach of disparate, manually customisable, ATs configuration.

\textsuperscript{4}Our web adaptations work on most standards-compliant pages, though the matter of standards compliance is acknowledged to be sporadic.
Table 9.1: Workflow trials overview: comparing phases of acceptance testing.

<table>
<thead>
<tr>
<th></th>
<th>Stage 1: pilot/exploratory</th>
<th>Stage 2: after refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num. users</td>
<td>≈10</td>
<td>≈20 (new users)</td>
</tr>
<tr>
<td>Reason for num. of users</td>
<td>Require expert feedback</td>
<td>Require varied backgrounds and preferences</td>
</tr>
<tr>
<td>Experience</td>
<td>Expert</td>
<td>Expert and novice</td>
</tr>
<tr>
<td>Purpose</td>
<td>Initial feedback</td>
<td>Comparison to existing AT approach</td>
</tr>
<tr>
<td>Under test</td>
<td>CAAR workflow</td>
<td>CAAR workflow</td>
</tr>
<tr>
<td>Control</td>
<td>N/A</td>
<td>Existing AT configuration methods</td>
</tr>
<tr>
<td>Primary data Scale</td>
<td>Qualitative Narrative</td>
<td>Qualitative Narrative; satisfaction</td>
</tr>
<tr>
<td>Secondary data Use</td>
<td>Learnt capabilities and preferences</td>
<td>Learnt capabilities and preferences</td>
</tr>
<tr>
<td>Use</td>
<td>Initial checking of process</td>
<td>Initial checking of process</td>
</tr>
<tr>
<td>Tertiary data Use</td>
<td>Logged events</td>
<td>Logged events</td>
</tr>
<tr>
<td></td>
<td>Testing capability store</td>
<td>Testing capability store</td>
</tr>
</tbody>
</table>

9.2.4 Question Design

Whilst the purpose may be to obtain information about how effective adaptivity as a paradigm can be in assisting people with minor-to-moderate impairments, it is vital to ensure that questions are not asked in a loaded fashion. Preferably questions should be asked in at least two different ways in order to help validate responses. Examples include: “Please describe, if applicable, how your computer usage changed” and asking questions such as “How would you describe your competence with the computer?” both before and after the study (not necessarily to rely on what is reported, but to check for any difference).

9.2.5 Stimuli

It cannot be prescribed that a person will experience capability barriers whilst participating in a study. This leaves two options, either: (1) simulate capability difficulties or (2) wait until such problems arise naturally. Option one may have to be employed in most studies and, likewise, presents two alternatives: simulate problems by altering the participant, or present the participant with taxing stimuli that may need to be adapted to be accessed.

Such stimuli would ideally include more “extreme” versions of stimuli with which the participant would normally come into contact: web pages; emails; documents. There is also the possibility of altering system settings to create the
illusion of an impairment, though this would require careful and sensitive planning to ensure it would not cause undue distress to the participant—and it would not be robust against the person using multiple devices, so is ideally suited to lab-based trials.

9.2.6 Early Lessons Learnt on Methodology

Early acceptance tests were carried out by Sus-IT staff and volunteer participants of the Dundee User Centre in order to gather feedback on the workflow and UI design. Some important sociological factors became apparent, with implications for the methodology used when assessing systems of this nature—i.e. those which would normally be deployed over a long period and learn about and work for the user gradually over time. In summary, adaptation initiative was as follows.

- Adaptations would be applied automatically when known to be required (triggered by user behaviour, in a similar manner to IBM’s WAT, first discussed in subsection 2.11.5).

- Similarly, very minor adaptations (e.g. mouse pointer size changes) that matched the user’s capability profile, would be automatically applied.

- When adaptations were thought to be required, but could have disrupted the user, they were suggested (indicated by a pulsing “help button”) via the UI shown in Figure 9.2.

Participants were given information-seeking and document-editing tasks to carry out, involving exaggerated content stimuli as described above: purposely poorly-designed web pages or documents. However, despite it being emphasised that the system, as opposed to participants, was under test, most people attempted to “muddle through” and complete the tasks as quickly as possible, rather than distract themselves with the as-yet unfamiliar help system. However, after having some time to experiment with the system after the tests, some participants felt that the adaptations on offer could have assisted them.

A number of lessons were learnt regarding methodology. It was already felt that the most appropriate way to test such a system—designed for long-term usage—would be longitudinally. However, it was felt important to gain feedback on the proposed interaction workflow before such a long-term test be conducted. In practice, it proved more difficult than anticipated to elicit this workflow feedback, due to the pressures of task-based assessments felt by participants. A goal-less trial period (with or without electronic or environmental stimuli designed to increase the likelihood of adaptations being invoked) was ultimately more useful.
It could be argued that the workflow for exploring adaptation options is technically deficient, at least in part, due to its failure to attract all users. However it is argued that the main cause of this is social: people expect to have to fit the device, rather than have the device fit them (see section 2.10). Once this cultural change happens—or, perhaps as a driver of it—the adoption of a standard UI convention that affords the concept of “adaptability” to the user will be needed.\(^5\)

The feedback gathered from this early testing was used to refine the UI in accordance with the target user group’s expectations.

**Simplification.** It had originally been the case that, when an adaptation was applied, an unobtrusive “feedback toolbar” would be displayed on-screen for a time. This contained options to explicitly accept, reject or highlight the adaptation in question. Some users did not notice the toolbar, whereas others felt that it was too intrusive.

Thus the toolbar was abolished and for the calibration “mini-games” and adaptations that were suggested, as opposed to automatically implemented, a “live preview” technique was used to show the effect of the adaptation across the whole system whilst the calibration or suggestions UI was still open.

**Consolidation.** Instead of presenting each calibration exercise/mini-game in a separate window, accessed via a menu, the mini-games and a menu listing them were folded into one window (as shown in Figure 3.16).

These changes meant that the user-visible system comprised only the calibration and adaptation selection windows, along with the ever-present “help button” on the desktop. There was (purposely) no user-visible UI in adapted applications such as Mozilla Firefox and Microsoft Word, as the adaptations are designed to be unobtrusive and controlled, if necessary, in a unified way via the “help button” workflow.

### 9.3 Longitudinal

In the long-term, Sus-IT will be carrying out longitudinal studies with participants from a range of groups connected with the project. The goals will be to continue to test the suitability and scalability of the approach, but also to collect data on capability fluctuations and accessibility barriers.

\(^{5}\)In much the same way that the now de-facto “feed icon” has been adopted to afford the concept of “periodic updates” [http://en.wikipedia.org/w/index.php?title=Feed_icon&oldid=483006644](http://en.wikipedia.org/w/index.php?title=Feed_icon&oldid=483006644).
The usability/accessibility hit-rate metric is expected to become a useful marker both for users (as adaptations and support materials that have helped similar users may be offered based on this data), developers (as users may give permission for anonymised usage statistics to be passed onto developers of applications, who can then assess the usability of those applications in different circumstances on a continual basis) and organisations providing support for such a system (as they will be able to target support resources accordingly).

9.3.1 Preliminary View on Results

At the time of writing, such longitudinal testing is underway; some initial findings have become apparent and, whilst not yet verified to be statistically significant, are summarised here for interest.

Perhaps most strikingly, most participants who self-identified as experiencing minor motor, hearing or vision difficulties, have explored the adaptivity system and discovered and continued to employ related adaptations. Many of these (mouse and double-click speed, for example) were already present in the OS but were inaccessible to the users. Some of these participants have explored other suggestions of adaptations (simplification of UIs or alterations to visual presentation of content, for example), but not necessarily continued using them.

A number of the participants who did not self-identify as having particular accessibility requirements have explored the calibration exercises and adaptations, but not always continued to employ the discovered adaptations. As only a limited range of adaptations could be supported, and capabilities and the environment fluctuate, this is expected—the key question is whether people either employ adaptations manually or return to the system for suggestions at times when they could be of help.

9.4 Developers

After initial workflow and user acceptance testing, Sus-IT will be carrying out more detailed longitudinal user trials, as well as working with partners and other AT developers to gauge their reaction to such a system. In order to prepare for this, several key milestones are planned and work towards these is in progress.

Three main types of developer stakeholders are relevant to this work; it is important to demonstrate the benefits of the proposed approach appropriately to each, as follows.

AT developers with which we have links will be invited to make an assessment of users of our system with their ATs.
Application developers will be invited to see the results and shown how to make their applications more DUET-aware and mutable.

Content authors (including website proprietors) will be shown the benefits of making their work more standards-compliant includes making it more adaptable and adaptive when the Sus-IT system is also in use.

9.5 Helping People with Real-world Problems

It has been discussed previously that accessibility barriers in the traditional, presentational, sense form only part of the barriers that users face in using systems [95]. Other work-packages of the Sus-IT project have investigated areas including reasons for technology abandonment and learning support needs which may be better-met to avoid abandonment. Summarising this work: key problems include mismatched mental models of how the technology works, as well as insufficient experience to find adequate support independently (e.g. tutorials on-line).

The central idea is to fuse context-sensitive help with adaptive accessibility. This will enable the user to seek help with the current task they are carrying out in a standard fashion (as with accessibility barriers currently), as well as having the help material (as with the application and task) rendered in an appropriate way. Further, the help can be tailored to the learning style and needs of the user: if videos are preferred to step-by-step textual tutorials, then videos (quite possibly from disparate locations, via the web) that have been found to help other users in similar situations, may be presented.

9.5.1 Testing Reasoning and Interaction

The kind of scenario described above may be used as an application to engage both users whose feedback is desired and developers, who wish to increase the market for their content, applications, services or support material.

We would create some tasks that users would generally enjoy; create help materials that are adaptable/replaceable and employ Frohlich’s detection and repair technique [39] (which would require us to develop our own abstract application). We then wait until the user asks for help, and can present adaptation suggestions as well as help suggestions, formatted appropriately for that user.

9.5.2 Advantages for Users and Developers

As has been discussed, a major drawback of adapted (not just adaptive) interfaces is that substantial differences between renderings for different people can cause
social or collaborative isolation. This has been partially addressed by mainstream products such as the iPhone, which affords people with sight loss the ability to appreciate spatial relationships amongst widgets and respond to requests to “try the icon in the top-left” by their collaborators.\textsuperscript{6}

In the realm of adaptive interfaces, another advantage of the approach proposed above, in which help material is marked up in reference to the abstract entities of the application is as follows: any step-by-step instructions can be altered to refer to the user’s adapted of the application if, due to their accessibility needs, this is markedly different than others’ versions. Further, the quickest—or most memorable for this person—way to access a given command within the interface can be ascertained and presented first (e.g. either traversing through menus, or employing shortcut keys dependent on the user’s abilities and preferences).

\section*{9.6 Summary}

Some key subjective and objective tests of the reasoning process, involving both end-users and developers, have been proposed. These are designed to ascertain the usefulness of the system in current real-world settings. Ongoing development work is being carried out to target forthcoming more abstract/mutable platforms, along with developers and content authors.

\textsuperscript{6}One person’s experience: \url{http://behindthecurtain.us/2010/06/12/my-first-week-with-the-iphone/} (retrieved on 20/12/2010).
Chapter 10

Conclusions

Many people are facing and will face a plethora of accessibility barriers as computing becomes more mobile, pervasive and personal. It has been argued that the current AT model is insufficiently flexible and requires extensive user awareness and is, therefore, unable to support these changes. A solution, namely that of employing small-scale adaptations when necessary to help alleviate minor-to-moderate impairments and transient accessibility barriers has been proposed and developed. The viability of this approach has been tested against key technical goals and the wider work of Sus-IT in performing large-scale and longitudinal testing of the approach has been defined and designed.

10.1 Contributions

Key contributions of this work are as follows.

- A review of the challenges involved in mainstreaming accessibility from the perspective of people with minor-to-moderate impairments.
- A high-level, problem-centred reasoning approach that operates on human terms and can be directed simply and by human input (largely passively).
- The ability to support people with multiple impairments or facing multiple, transient, accessibility barriers, by matching them to appropriate adaptations; allowing users to benefit from adaptations of which they were previously unaware.
- Support for collaboration of a number of different types between users with various impairments and/or preferences.
- Relatively low-effort compatibility with existing ATs and research projects on adaptivity.
• A simple interface to dealing with adaptations for platform vendors and application developers.

The remainder of this chapter discusses both ways in which the work can be extended as well as other possible applications of the reasoning process that are of interest.

10.2 Extending the Theory

There are a number of directions in which the theory developed could be extended, as follows. Avenues for extending CAAR temporally were discussed in chapter 7.

10.2.1 Scale

Designing not for two or three users, but hundreds, or thousands, at once would focus concentration on developing efficient ways to store, access and update users’ data—capabilities, preferences and context. If adaptability and adaptivity is to be embedded in all ICTs then the system that operates it, like the Domain Name System (DNS) of the Internet, must be ubiquitous, efficient and transparent. The current model of local control systems managing a group of devices could in theory scale quite well; it is de-centralised (on a global basis) so in effect reliable. However a challenge would be federating users’ data amongst such systems, as to be useful it would have to be globally accessible. These are, however, challenges that the semantic web, upon which the present user profiling technique is built, was designed to address. Further, the GPII project is investigating these challenges and, as has been noted, working towards becoming part of the infrastructure is a natural goal.

How specific or broad do capabilities need to be in order to be of use to the system and, therefore, users? How many capabilities can reasonably be specified? The examples given here have been relatively small-scale to prove the concept and demonstrate tractability, rather than trying to represent $n$ years’ worth of stored capability data. In fact, the length of time that such data may be needed is another question—if historical interest is minimal, then perhaps this is not a huge problem, but it would be interesting to see what can be gleaned from tracing capabilities back in time. One means to avert huge data storage requirements would be to take the approach of specifying only what is different about this user—the components and capabilities possessed as they deviate from the aggregated values across the population.

The possible benefits of “crowdsourcing” accessibility suggestions should not be overlooked. If the problem is, due to this work, now expressed at the right level
then sufficient data mining effort should give rise to sensible recommendations for users based on the activities of their peers in similar situations or facing similar capability barriers.

Another avenue would be to test the usability/accessibility metric on a large scale and assess how useful developers may find it. Many already collect telemetry data from applications (web browsers being a relative hotbed for this presently, though office applications and OSs in particular often make use of less user-facing data gathering, to fix driver bugs).

10.2.2 Cognitive Adaptations

This thesis has used capability-based reasoning mostly in the realm of purely presentational adaptations. Cognitive adaptations are under-researched. However they are generally the most prevalent kinds of barriers people face. Having a mental model that does not match a technology can have a profound impact on usability and accessibility. A system to help people cope with such barriers could provide an avenue towards mainstreaming accessibility.

Users could be facing instantaneous problems such as from remembering how to engage a certain process within an application (“How do I?” or “Where is?”). There may also be incompatibilities between how the workflow of a system is arranged and the user’s expectations. For examples not meet their expectations, such as a set-top box categorising media by data type (pictures, videos, audio) as opposed to an event-oriented view exemplified by “Our holiday in Blackpool”. With the progressing shift to more malleable applications, delivered over the web, there is an unprecedented opportunity to encourage the mainstream to expect more flexibility and developers to provide the APIs to support this.

The cognitive area is also a great challenge because notions such as compatibility between adaptations become less of a presentational matter and much more of a potential absolute accessibility barrier—if someone cannot understand a system then they certainly cannot use it. One such case, the novice user interacting with the vision-impaired expert from subsection 6.12.2 was genuine but had a convenient solution in which the other user could drop a constraint. However, orchestrating communication between people unable to take approach is a compelling challenge and an increasingly important one.

10.3 Alternative Uses

There are number of areas in which capability-based reasoning could be applied that do not fall into typical ICT scenarios discussed here.
• Recommending products or services (noting the serious ethical considerations). E.g. modelling differences between a 50” standard-resolution and a 32” “high-definition” television for a person with given capabilities. Likewise as applied to the feature sets of smartphones.

• Supplementing the persona approach used by some developers—real data could be used to populate the personas. Helping to test products more cheaply by catching some of the most basic errors before testing with real people.

• Matching participants in online games or virtual environments, either for competitive or co-operative purposes (emphasising what people can achieve).

10.4 Summary

The capability-based adaptive accessibility approach is proposed as a means to both improve the accessibility of mainstream systems for people who may not regard themselves as disabled, or those experiencing capability fluctuations brought about by the ageing process, as well as a way to improve the provision of traditional ATs for people with more severe disabilities.

Whilst the approach is new, it builds upon a wealth of previous experience and artefacts developed across academia and industry. Though there are many possible enhancements and future extensions, CAAR is already being used as a central tool of the Sus-IT project in order to improve the autonomy of older people using ICTs.
Glossary

**AAL** Ambient Assisted Living. 53

**AI** Artificial Intelligence. 11, 53

**API** Application Programming Interface. 26, 39, 42, 147, 165, 209, 229, 233, 272

**AT** Assistive Technology—Hardware or software designed to allow a user to overcome an accessibility barrier (such as lack of sight or hearing). ii, 1–3, 19, 21, 24, 26, 27, 29, 30, 32, 34, 35, 37–43, 46–48, 50, 51, 54, 58, 59, 67–69, 74, 104, 122–124, 128, 170, 178, 186, 211, 213, 215, 216, 220, 221, 224, 227, 230–232, 271

**AT-SPI** Assistive Technology Service Provider Interface—A cross-platform, toolkit-neutral library that is used to expose accessibility information from a widget toolkit to an AT via its standard interface. 39, 74

**AUIML** Abstract User Interface Markup Language. 17

**CAAR** Capability-based Adaptive Accessibility Reasoning—The theory developed in this thesis. 62, 74, 82, 89, 90, 127–129, 131, 133, 144, 149, 151, 155, 163, 183, 185, 186, 192, 199, 209, 211, 215, 216, 221, 228, 230, 257

**CC/PP** Composite Capability/Preference Profiles—A World Wide Web Consortium (W3C) standard that aims to provide a way for the capabilities of devices and web user-agents to be described, in order that content be adapted for them specifically. Related to UAProf. 66, 234, 268

**CLI** Command-Line Interface—An interface that requires the user to type in commands and generally produces only text output. 10, 104

**CRUD** Create, Read, Update and Delete—The four essential operations that may be performed on data (usually referring to data held electronically). 233

**CSCW** Computer-Supported Co-operative Work. 19, 53
DBMS  Database Management System. 8

DOM  Document Object Model. 34, 44, 262, 272

DPI  Dots Per Inch. 202, 203

DUET  Device, User, Environment and Time—The set of capability data (in human terms) required to describe a context-agnostic instance of an accessibility problem. 2, 34, 57, 58, 64, 65, 67, 69, 74, 84, 102, 185, 212, 225

DUETS  Device, User, Environment, Time and Situation—The full set of capability data (in human terms) required to describe a specific instance of an accessibility barrier. 24, 50

GPII  Global Public Inclusive Infrastructure—A project to create ubiquitous means of discovering and delivering ATs. http://gpii.net/. 47, 54, 211, 228

GUI  Graphical User Interface—Historically often referred to as a WIMP interface. 6, 7, 10, 15, 17–19, 22, 38, 39, 42, 45–47, 50, 51, 60, 68, 99, 102–104, 173, 190, 235, 255, 271

ICF  International Classification of Functioning, disability and health. 28, 66, 86, 89, 160, 202, 211, 262, 270

ICT  Information and Communication Technology—The more modern variant of “IT” with “Communication” added to reflect the growing trend of using computing technologies for social means. 20, 21, 23, 53, 78, 228–230

IE  Information and Environment. 133, 144, 146, 155, 168

KBFE  Knowledge-Based Front-End. 9

MSAA  Microsoft Active Accessibility—A standard API developed to allow ATs running on Windows to interrogate applications for accessibility information. 35, 39, 74

NDA  New Dynamics of Ageing—A cross-council funded research programme in the UK—the largest to-date to be carried out on ageing; http://newdynamics.group.shef.ac.uk/. ii, 1, 234

netbook  A modern type of sub-notebook computer with a small screen (usually 8–10” diagonal), often running Linux or a low-resource version of Windows and having solid-state storage as opposed to a hard disk drive. 15, 43, 255
OCR  Optical Character Recognition. 39, 93

OE  Operating Environment—Effectively the “shell” presented by the OS that the user interacts with. An OE may not necessarily be part of the OS—it could be an add-on program—and, on its own, it may not be able to fulfil all of the (low-level) functions that the OS does—namely protection and sharing of resources. 39, 40, 68, 232

OS  Operating System—A low-level component of a computer system that is responsible for the protection and sharing of resources amongst user programs. Contemporary OSs often provide simplified graphical abstractions of the computer system to the user in order to allow them to focus on carrying out their tasks rather than managing the operation of the system. They often also provide a number of APIs for application programs to build upon, abstracting the lower-level details of using the hardware from programs. 10, 15, 26, 33, 42, 50, 52, 72, 95, 103, 128, 148, 155, 160, 166, 173, 179, 200, 201, 209, 215, 224, 229, 232, 233, 257, 271

OWL  Web Ontology Language—This is a higher-order language for expressing knowledge (including classifications and relationships between entities) and builds on the lower-level Resource Description Framework (RDF) standard. 268, 270

PAN  Personal Area Network—A very small-scale network, consisting of devices largely expected to belong to one person (as opposed to a LAN, which is expected to contain devices belonging to a corporation or institution). Typically short-range wireless standards such as Bluetooth are used for communication. 18

PDA  Personal Digital Assistant. 15, 45, 51

PUC  Personal Universal Controller—A device that can generate an appropriate interface for interacting with a given appliance in the home, based on an abstract service description provided by the appliance. 17, 50, 51, 119

RDF  Resource Description Framework—The low-level building block of semantic web standards from W3C; RDF provides a way to express knowledge as simple statements in the form of subject-predicate-object triples. Many other standards are layered on top of this. 233, 268, 270

REST  REpresentational State Transfer—A conceptual approach to designing data-driven applications in which the interface and data semantics are separated, data is exposed via URLs and the standard Create, Read, Update
and Delete (CRUD) API is used to communicate with the application over a stateless protocol—namely HTTP. This allows the application to be accessed in a variety of means, simply and reliably over a network. 165, 166


**SaaS** Software as a Service—Software be provided not on dedicated media or as a discrete application package for installation over the network, but as a service, which is hosted and often called remotely and charged for by rate of use rather than discretely on each product iteration. 272

**SBML** Style-Based Markup Language. 17

**STT** Speech-To-Text. 93

**Sus-IT** Sustaining ICT use to promote autonomy—A multi-disciplinary research project funded by all five UK research councils as part of the NDA programme. See [http://sus-it.lboro.ac.uk/](http://sus-it.lboro.ac.uk/). ii, iv, xiv, xvi, 1, 2, 4, 5, 25, 44, 47, 49, 55, 57–59, 63, 69, 72, 74, 77, 78, 81, 82, 90, 101, 126, 128, 133, 155, 157, 160, 162, 174, 186, 209, 212–214, 216, 218, 222–225, 227, 230, 261, 267, 268

**TRIALS** Time, Redundancy, Information, Aesthetics, Layout and Sharing—The set of global indicators that enable monitoring of the effects of an adaptation at a higher level than in just one channel. 139, 146, 147, 151, 154, 165–168, 170, 172, 186, 187, 205, 209, 275

**TTS** Text-To-Speech. 18, 19, 39, 40, 44, 45, 48, 50, 51, 97, 107, 124, 137, 140, 144, 174, 215, 276

**UAProf** User Agent Profiles—A W3C standard related to CC/PP. 66, 231


**UIML** User Interface Markup Language. 17

**UIMS** User Interface Management System. 8, 9, 12, 16, 98

**UN** United Nations. 28

**URC** Universal Remote Console. 17
**VM** Virtual Machine. 12

**W3C** World Wide Web Consortium. 231, 233, 234, 268

**WAT** Web Adaptation Technology. 42, 44, 45, 50, 75, 155, 222

**WHO** World Health Organization. 28, 66

**WIMP** Window, Icon, Menu, Pointing device—A type of GUI that expresses the computer system in terms of analogies such as the “desktop” and “documents” and uses the mouse extensively for interaction. 7, 10, 15, 17, 42, 54, 102–104, 232

**XAML** eXtensible Application Markup Language. 17, 47

**XIML** eXtensible Interface Markup Language. 17

**XSL** eXtensible Stylesheet Language—The W3C standard for specifying documents that, when passed to an XML processor along with an input XML file, perform a transformation on the input XML file. An example of such a transformation may be an XHTML-to-plaintext converter. 16

**XUL** XML User interface Language. 17, 40, 47

**ZUI** Zooming User Interface. 54, 104
References


REFERENCES


REFERENCES


REFERENCES


REFERENCES


REFERENCES


Appendix A

Contrasting Usability and Accessibility

This appendix describes the usability and accessibility concerns of two prominent user interface devices. It discusses the growth in prevalence of accessibility barriers and how these are affecting more people.

A.1 The Office Ribbon

The ribbon replaces the traditional collection of menus and toolbars in Microsoft’s Office suite of applications. It follows from previous attempts at interface reform, one of which was analysed in subsection 2.3.1. As discussed below, the ribbon has brought a range of usability improvements (and challenges) but also brings with it some considerable accessibility barriers. The currently-available version of Office is “2010” but this section also considers changes made by the previous version; “2007”—the first to implement a “ribbon”.

A.1.1 Usability Achievements

From a usability point-of-view, a considerable amount of research and development effort went into determining the problems with the existing interface—complexity, clutter, poor organisation of commands—and developing the solution: a tabbed toolbar known as “the ribbon” [57] that, as with the toolbars that preceded it, sits along the top of the application’s window, above the document and is depicted in Figure A.2.

One key feature of the ribbon in contrast to the previous attempt at interface reform (“personalised menus”) is that it is only slightly adaptive: though some tabs appear when certain objects, such as pictures, are selected, the core tabs
APPENDIX A. CONTRASTING USABILITY AND ACCESSIBILITY

Figure A.1: Word '97 (top) and 2003 (bottom) default toolbars.

Figure A.2: Two versions of the Word ribbon: 2007 (top) and 2010 (bottom; with improved contrast). From [113].

are always visible and none of the individual commands (represented by their toolbar icons) move, allowing the user to achieve high efficacy by using motor memory, which was not possible when using personalised menus as they were implemented.¹ This solves the problems of unpredictability, whilst also allowing the interface to remain somewhat context-sensitive and therefore generally less cluttered, as irrelevant commands need not be presented. Overall, common tasks are generally equally or more readily accessible by mouse (keyboard shortcuts are still usable), as shown by Table A.1, though some commands require more steps to reach in this manner.

In the table, the most popular commands are listed in order first [56], followed by some further popular commands [74] and then a selection of other available functions. Note that keyboard shortcuts are not listed as they have remained the same (and users still make use of buttons such as “Paste” despite such shortcuts being available [56]).

Perhaps the most important aspect of the ribbon is the re-organisation of commands that it has brought about. By way of example, the designers list several functions that were available in Word 2003 [57], as follows.

- Find out the current number of words
- Turn on speech command and control

¹As was pointed out in subsection 2.3.1, split menus could have been used to resolve this problem.
Table A.1: Comparison of mouse clicks needed to achieve actions in Word 2003 and Word 2010 (when the user is on the “Home” tab of the ribbon and otherwise).

<table>
<thead>
<tr>
<th>Task</th>
<th>Version</th>
<th>n</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paste</td>
<td>2003</td>
<td>1</td>
<td>Toolbar button</td>
</tr>
<tr>
<td><em>(also: Save, Copy, Bold)</em></td>
<td>2010 (Home)</td>
<td>1</td>
<td>Paste</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>2</td>
<td>Home → Paste</td>
</tr>
<tr>
<td>Undo</td>
<td>2003</td>
<td>1</td>
<td>Toolbar button</td>
</tr>
<tr>
<td></td>
<td>2010 (Home)</td>
<td>1</td>
<td>Quick-access button</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>1</td>
<td>Quick-access button</td>
</tr>
<tr>
<td>Superscript <em>(also: Subscript, Strikethrough)</em></td>
<td>2003</td>
<td>5</td>
<td>Select, Format → Font → Superscript → OK</td>
</tr>
<tr>
<td></td>
<td>2010 (Home)</td>
<td>2</td>
<td>Select, Superscript</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>3</td>
<td>Select, Home → Superscript</td>
</tr>
<tr>
<td>Text style</td>
<td>2003</td>
<td>2</td>
<td>Select style in drop-down list</td>
</tr>
<tr>
<td></td>
<td>2010 (Home)</td>
<td>1/2</td>
<td>Top 3 styles visible; others via list</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>2/3</td>
<td>Home → Top 3 styles visible; others via list</td>
</tr>
<tr>
<td>Insert table</td>
<td>2003</td>
<td>3</td>
<td>Table → Insert → Table</td>
</tr>
<tr>
<td></td>
<td>2010 (Home)</td>
<td>2</td>
<td>Insert → Table</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>1/2</td>
<td>Insert → Table</td>
</tr>
<tr>
<td>Spell Check</td>
<td>2003</td>
<td>1</td>
<td>Toolbar button</td>
</tr>
<tr>
<td></td>
<td>2010 (Home)</td>
<td>2</td>
<td>Review → Spell Check</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>1/2</td>
<td>Review → Spell Check</td>
</tr>
<tr>
<td>Thesaurus</td>
<td>2003</td>
<td>3</td>
<td>Tools → Language → Thesaurus</td>
</tr>
<tr>
<td></td>
<td>2010 (Home)</td>
<td>2</td>
<td>Review → Thesaurus</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>1/2</td>
<td>Review → Thesaurus</td>
</tr>
<tr>
<td>Quick Print</td>
<td>2003</td>
<td>1</td>
<td>Toolbar button</td>
</tr>
<tr>
<td></td>
<td>2010 (Home)</td>
<td>3</td>
<td>Office menu → Print → Quick print</td>
</tr>
<tr>
<td></td>
<td>2010 (other)</td>
<td>3</td>
<td>Office menu → Print → Quick print</td>
</tr>
</tbody>
</table>
• Create a SharePoint Document Workspace
• Print Envelopes
• Open the Visual Basic Editor
• Turn on hyphenation
• Merge the contents of multiple documents
• Start a web conference
• Tweak AutoCorrect settings

All of these functions were found on the “Tools” menu in the previous version. Unless a user already knew this, they would undoubtedly struggle to find the settings—and even if they were not new to the software, they would have to re-member the illogical hierarchy of commands. Under the ribbon scheme, commands are grouped according to the task, for example: document review; mail-merging or software development.\(^2\)

According to the evaluation presented by the designers, the interface has been well received. Though many people may be initially averse to the significant organisational change, those that have used the interface for some time seem to have developed a high rate of positive acceptance—over 80% of participants in a 2-month usability study strongly or somewhat agreed that the new interface made it easier to create professional-looking documents and was more fun to use than the previous one [57].

Since the original ribbon was released as part of Office 2007, several improve-ments have been made. The main usability improvement is that the ribbon is now customisable: new tabs may be added and commands can be added to exist-ing tabs. This may be thought of as a manual version of Gajos’ “fixed adaptive area”, to which an individual user’s frequently-used commands were automatically added, thus improving performance, but not altering the rest of the interface.

A.1.2 Accessibility Barriers

Unfortunately the same positive outcome has not been the case in terms of accessi-bility. As a very simple example, consider the artistic style of the “2007” ribbon,

\(^2\)In the interests of full disclosure, it should be noted that the development toolbar is disabled by default and that a small number of the options listed—speech and collaboration—are no longer features available in Word “out-of-the-box”.
in comparison to the more basic rendering used in earlier versions of Office (par-
ticularly the Office '97 toolbars, shown in Figure A.1). Though some users enjoy
the artistic style, others have found it distracting and the icons unclear relative to
previous versions of Office (see the comments in [113]). In fact, this problem has
been so prevalent that the “2010” ribbon has a 5:1 contrast ratio and simplified
artwork, as shown in Figure A.2 and discussed by the developers [113].

Further, the inability to orient the ribbon vertically instead of horizontally
presents barriers for users of widescreen displays (particular the small ones now
common in netbook computers). The effect of this is that a large proportion of the
display is taken up by the ribbon [123]. Naturally, orienting the ribbon vertically
would mitigate any “motor memory” the user may have acquired when using the
horizontal rendering, but it is likely that a significant number of users would prefer
to have the choice, rather than to struggle to edit documents on devices with small,
wide screens.

For users of screenreaders, the ribbon appears to present an ideal opportunity
to ease the process of exploring the available commands (shortcuts may be used, as
always, to activate specific commands, which may prove quicker for screenreader
users). The logical way to present the structure of the ribbon would be as a list of
lists (or a tree): each ribbon tab would be a node at level 1; each section of each tab
would be a node at level 2 and each command (represented in the GUI by a toolbar
button) would be a node at level 3 and could be represented in speech or Braille
as a linear list of commands in each section of each tab (with the order fixed and
with the most commonly-used commands first). This would mirror the graphical
structure yet remain simple to navigate. Unfortunately, the reality is that users
of some screenreaders are required to navigate the ribbon two-dimensionally using
the arrow keys (whilst others are given the option of cycling through the controls
in some sort of order). This seems to be a retrograde step for accessibility as it
compels the user to be aware that there is a two-dimensional representation of the
ribbon—such knowledge is unlikely to be of benefit to them, but complicates the
navigation process.

\footnote{which were very similar in appearance to the Office 2000 toolbars and similar to the Office
XP toolbars}

\footnote{It is possible to “minimise” the ribbon so that just the top-level tabs are visible—however
this means that none of the commands in the ribbon can be accessed directly and the whole
ribbon must be made visible to select and activate one, unless keyboard shortcuts are used.}

\footnote{This was ascertained from both documentation for a screenreader (http://www.
yourdolphin.com/tutorials/index.asp?id=120) and confirmation from a blind user of Of-
lice 2007, via a different screenreader.}
A.1.3 Contrasts and the Prevalence of Accessibility Barriers

The ribbon has clearly benefitted from significant effort in the area of usability—both at the design and testing stages. It appears to keep most common tasks quick, make some other common tasks quicker and vastly improve the process of command discovery by categorising commands logically. Unfortunately, it has also introduced some accessibility barriers (at least one of which appears to have been revised for the “2010” version).

Three negative aspects of the new UI have been presented as “accessibility barriers” as opposed to usability problems. This is in accordance with the definitions given in the literature review; justification for the classification for these particular examples follows.

The artistic style presents an accessibility barrier because it affects some users’ abilities to perceive the interface—what the meaning of the toolbar buttons is. The interface may only be used (and possibly usable) if it can be perceived.

Ribbon position is also an accessibility barrier, because it affects users who are effectively impaired by way of the device they are using to access the software. The designers did not take this route of access into account and, thus, did not provide an alternative—or adaptable—UI for this situation.

The navigation complexity is an accessibility problem because it affects users who are using a “replacement” interface via a different modality (and of a lower dimension). It compels these users to be aware of the standard, higher-dimension UI, which increases cognitive load.

Clearly these barriers may affect different users, in different situations, by varying amounts. Though they are classified as accessibility barriers for the purposes of the present research, as is pointed out in subsection 2.9.4, a more suitable metric is needed for use from the users’ perspective(s). Such a metric is developed in chapter 4.
Appendix B

Runtime System Architecture

This appendix continues the branch of design work that began with Figure 3.12, which provided a very high-level overview of a runtime system from a central viewpoint. It describes the layout of a CAAR-based run-time system in progressively more detail. Of particular note are the expected usage scenarios and design assumptions and goals.

The focus in this appendix is on the “outer” parts of the system; those that interact with devices, the user and environment. It is expressed at the level of the OS processes that make up the system and the types of communication needed between these processes.

B.1 High-level Usage Scenarios

The usage scenarios for the system can all be built from two primitives, shown in figure B.1. Some slightly more complex scenarios are depicted in figure B.2 and some further expected use-cases are shown in figure B.3.

There are two cases that deserve particular attention. The first is a matter of determining which user is providing input when multiple users are using a particular device. In a collaborative environment, a user may be joined by colleagues to demonstrate work, or work collaboratively on a given task. In such a situation, it is assumed that the user providing the input to a device is the primary user of that device.

The second case of interest is highlighted by scenarios C and E and occurs when an individual user is using more than one device. When the user calls for assistance, it is necessary to work out which part of the application interface—and, thus, device—the user requires assistance with.

Finally, Figure B.4 shows an example scenario involving multiple environments. In section B.3, the relationships in terms of data and principal execution flows between an individual Controller, Device and User (connected by dashed lines in...
Figure B.1: The two usage scenario primitives.

Figure B.2: Examples of more complex usage scenarios.

Figure B.3: Further examples of more complex usage scenarios.
Figure B.4: An example multiple-environment usage scenario (the dashed lines connect the nodes of interest in the following, detailed diagrams).

Figure B.4) are presented. The following section gives some design assumptions and introduces some necessary supporting processes for the run-time.

## B.2 Detailed System Description

As above, this section gives the basic design assumptions of and introduces processes needed to support the run-time system.

### B.2.1 Basic Assumptions

For the run-time surrounding the reasoning process, the following technical assumptions were made.

- The path from User to Application (and vice-versa) is critical and will require realtime communications.

- Communication on the local network is low-latency, reliable and cheap.

- Communication to external network hosts could be adequate, but is potentially high-latency, unreliable and expensive.

- Caching may be used to alleviate pressure on remote systems (in the style of HTTP caching).

Ideally, there should be a local-network path between User and Application. However, in a world of increasingly distributed and service-oriented software, this may not always be possible. It is in this case that the continued—enforced, in this case—separation of user interface and core application code is beneficial, as the abstract interface and (portable) code driving it may be cached locally, whilst the (still remote) application provides processing for long-running tasks. Spiritually,
this could be implemented in the same vein as the contemporary AJAX+REST style of design pattern on the Web. Through the requirement for a machine-readable abstract interface specification, however, the contemporary pattern is generalised.

A key design goal for any runtime system of this nature is to minimise the number of processes which require—and, particularly, store—data that the user may regard as personal (i.e. widget usage and adaptation activity). This leads to the concept of some processes needing to be trusted by the user, which adds to the argument for keeping these processes (such as the Controller) as local as possible and having transient processes, such as those monitoring the user, not store any records of their own.

One process that both must keep records and is likely to always be remote is that process which stores user profiling information on behalf of the user. This process would likely need to be globally accessible for each user, so deserves particular attention in relation to privacy security matters.

**B.2.2 Components of the System**

As discussed, several process are involved at device-level in the runtime system. These are recapped and, where necessary, introduced below. Diagrams provide visual counterparts to the following descriptive sections. The figures show how the processes would likely be arranged in a number of contemporary desktop/portable and embedded usage scenarios.

**B.2.2.1 Key Actors and Processes**

**Devices and T-devices** as described elsewhere.

**The User** is the principal user of a given device. Multiple users may use any given device, though it is likely only one will be providing input at any given time. Output from the device must be suitable for all users of the device; these sections, however, concentrate on the input pathways from the user.

**The Environment** is the local vicinity in which users and devices can be found. Within an environment, users may communicate with each other directly; across environments they may not. Each environment may have its own Controller process.

**Controller processes** are responsible for managing the presentation of interface and output to users of devices in the local environment. They maintain an abstract interface tree (for non-legacy applications) and are aware of the pertinent properties of uses and devices currently active in that environment.
A Controller may be present for each office within an organisation, or in each home. It is the Controller that carries out the fundamental reasoning processes which are developed in this thesis.

B.2.2.2 Device-local Processes

The Device OS is used to provide notifications of certain events of interest (such as user-executed adaptations).

Legacy Applications must be supported, likely through the use of modification adaptations. On some platforms it may be possible to record at least partial usage statistics for legacy applications (often using the accessibility APIs).

Adaptive Application UIs are presented which correspond to the relevant (for this User and Device) parts of the overall Applications’ interface.

The Monitor process orchestrates the recording of data, rendering of adaptive UIs and, through plug-ins (not shown on the diagrams that follow, but shown in Figure 3.12), execution of both global and application-specific adaptations. All of these activities are carried out on the instruction of the group Controller process.

A Monitoring process exists for each active session on a given device. In the case of a desktop computer, which could have multiple users (and multiple users active at once), there would be one such process for each logged-in user.

It is important to note that if multiple people are working next to each other on one device, there would likely be only one Monitor process (corresponding to the User whose account is currently logged in). In this case, there would need to be a mechanism for informing the controller of the presence of other users. Methods to accomplish this are out of the scope of the current work, but are being investigated by Sus-IT.

Adaptations are the processes that actually perform adaptations to the system. They are plug-ins for the Monitor, which is capable of launching them as and when required by the Controller. Adaptations may have the following components.

The Affector is the part that carries out changes to the system and will be highly platform- and perhaps application-specific.

The Trigger watches (via messages subscribed to via the Monitor) for local events/data of interest that may indicate that the services of this ad-
aptation may be required (e.g. observing mouse input to detect certain motor control problems).

**Metadata** to describe the adaptation according to the classification (see Appendix D) so that the Controller can reason about it effectively.

**Technical effects** information. E.g. the use of full-screen magnification at a linear scale factor of two implies that all font point-sizes will be twice as large as before the adaptation was invoked.

Standard technical units, such as those given in [64] would be used. The effects on these may be computed by using a technical model in the adaptation, or simple modifiers such as scaling (as would be the case with full-screen magnification).

**A Model** used to predict human capability ranges, based on available technical information.

The standard capability classification (ICF, possibly extended) is used.

The part of an adaptation that actually affects a change on the user’s system—the Affector—may be present as a plug-in for another application on the system and, thus, run in that application’s process. Examples may include adaptations for specific applications such as Firefox or Word, which require access to the DOM or similar for those given applications. In such cases, the Affector must communicate with the adaptation plug-in inside the Monitor in order for it to receive instructions from the Controller and provide information to the Controller and any other interested adaptations on the system (via the Monitor’s blackboard).

**Feedback** on proposed and executed adaptations needs to be gathered from the user; an additional process is shown on the following diagrams that corresponds to this task.

**B.2.2.3 Abstract Application Serving**

The applications discussed here are the forecast more abstract and adaptation-friendly (as opposed to legacy, adaptation-unaware) applications.

**Application libraries** provide a catalog (using the “known place” principle often applied in networking (to make finding services easier) of available adaptive applications. The libraries supply the abstract interface and any (portable) code to drive it to Controllers.

It is envisaged that a particular university, corporation or user-group may host specific libraries for its users.
Applications actually carry out the tasks requested by the user and are effectively shielded from the Controller (and User(s)) by the library.

B.2.2.4 User Storage and Analysis

Storage and analysis is required on a long-term, per-user, basis for data collected by the monitoring process. This is then used by the Controller to improve its reasoning regarding assignments of Users to Devices and the rendering of interface and content elements.

The Monitor processes on each Device send the data recorded to Controllers during the operation of the system. At particular times (e.g. when a User logs off a device), this data is packaged and sent on behalf of said User to the Storage process, which is assumed to be remote.

B.2.3 Types of Reasoning

There are three types of reasoning that are carried out by the system as a whole.

Control is the normal style of reasoning where information on adaptation and other activity is passed to the Controller and, in conjunction with user data (predictions, model, preferences), decisions are made as to which adaptations may be most appropriate at any time and whether these should be applied automatically or presented to the user only when the user requests help.

This is carried out continuously as long as users and devices are connected.

Analysis is the process by which the logged raw data (adaptation activity, feedback and calibration data) held in storage on the user’s behalf are analysed and the user model is formed—helping the Controller to answer questions such as what the most appropriate course of action is when the user finds there is too great a volume of information in, for example, a web page.

This is carried out periodically, on behalf of each user (though it is possible that a Storage service could aggregate across its whole user population anonymously, for the purposes of arriving at a useful starting model for a user).

Reflex is where an adaptation’s trigger condition is met by means of it receiving messages from the Monitor (i.e. local events of interest have occurred and the adaptation believes that it is able to help the user).

This is carried out spontaneously by an adaptation in response to information received locally from other adaptations.
B.2.4 Process Responsibilities

The Monitor acts as a blackboard for the adaptations on the local system and conveys relevant data to the Controller and executes the Controller’s commands in terms of launching adaptations.

The Controller refers to information from all relevant Monitors, user models and classifications to reason about which adaptations should be proposed or activated at any time (and whether further calibration data is needed).

The Storage server stores data logged by the Monitor(s) and fed to it by the Controller on the user’s behalf.

The Analyser carries out the reasoning processes centrally on behalf of users.

B.3 Detailed Usage Scenario Diagrams

The data and execution paths shown in the diagrams below are as follows.

Interface (Application to User) flows are shown in black and represent the abstract interface provided by an application on its journey to being rendered for a given user, in a given manner on a particular device. Note that the interface could be split amongst multiple users and devices.

Interface Event (User to Application) flows represent interface actions carried out but the user being conveyed to the application (and, eventually, recorded in the user’s data store) and are shown in red. Again, flows to one controller and application can come from multiple users and devices.

Focus flows, purple, denote when the user switches between applications.

Profile data is required by the controller to reason about users and devices; these flows are shown in blue on the diagrams.

Adaptation flows represent when the user or environment has requested or caused a change to the system. Such changes may result from fluctuating weather conditions, the user making an adaptation to the operating system of the device (such as contrast, volume or text size) or the user accepting or rejecting an adaptation proposed or executed by the system (controller).

Figure B.5 shows a contemporary desktop/portable computer scenario. Figure B.6 shows this arrangement with a direct-rendering shortcut, which would allow the application to communicate more rapidly with the computer upon which it
is apparently (to the user). This shortcut could be implemented in much the same way as direct rendering in X Windows. Figure B.7 again shows the contemporary scenario, but with direct-rendering shortcut and controller running on the local machine.

Figure B.8 depicts a situation in which an embedded device is being used. Examples of this include set-top boxes and smartphones. Figure B.9 demonstrates a slight alteration of this embedded scenario, in which the controller on the local device (as may occur if the user is away from all of their other computing devices).

### B.4 Further Work

In parallel to the development of the theory behind the system, the following areas have been investigated by co-researchers, Ph.D. and project students related to
Figure B.7: Local processes: composite diagram.

Figure B.8: Embedded with external Controller: composite diagram.
Figure B.9: Embedded with internal Controller: composite diagram.

Sus-IT; these included Yunqiu Li, Adam Cox and Kirsty Smith.

- Security of stored data and log-on credentials. A challenge-response approach is in development for communication with the data store.

- Identification of users in a collaboration on one machine. Users additional to the logged-in user on the machine need to be identified. An approach using standards such as ad-hoc networking is in development.

- Inter-process (and inter-machine) communication protocol. Events of interest to the adaptivity system must be transmitted to relevant agents. A protocol has been developed to achieve this.
Appendix C

Capability Classification in Practice

This appendix details the real-world capability and component classification developed for Sus-IT’s run-time system.

C.1 Capability and Component Classification Design

The design of the standards was modelled after the W3C’s approach to “semantic web” standards such as RDF and, layered on top of this, Web Ontology Language (OWL). The main reasons for this included: the layered approach; the fact that both of these standards are concerned with modelling real-world phenomena, as opposed to other layered standards such as network protocols, which are primarily concerned with managing the internals of a computer system and the fact that standards expressible as RDF can be distributed, shared and easily extended using a variety of existing tools and techniques.

A key facet of many W3C standards is extensibility. This is often achieved by defining a standard (such as CC/PP) in two parts: a schema, which describes the vocabulary used in the standard’s domain and a reference document that is constructed using the vocabulary defined by the schema. In the case of CC/PP, however, the goal is to describe device capabilities in device terms. This is a needed part of the reasoning stack, but it is at a level different to that of the reasoning processes developed here.

In order to reason about and with human capabilities, the following components were deemed necessary and sufficient.

A schema that provides the vocabulary to describe a capability, its properties
and the structural (anatomical) links between capabilities in a given species.

A capabilities “map” that enumerates and describes all possible (human) capability and their associated properties and the normal anatomical links between capabilities for the human species. This is required in order to express the fact that certain capabilities are related (e.g. aggregate visual acuity is formed from both eyes and that two eyes are required for stereoscopic vision) and some are redundant (e.g. both hands are intended to have similar functionality).

A corresponding schema and map for classifying components—those organs or structures that afford capabilities—in a machine-readable way. Such a standard would enable users and their profiles to indicate preferences as to which components be used in certain situations (such as being left- or right-handed).

Instances of the capabilities map—effectively user profiles—can then record, for a given person, the values of the capabilities that individual possesses. It is important to note the following.

- The schema does not describe any capabilities; it only supplies the vocabulary to do so.

- The map enumerates all possible human capabilities and structural links between them. These are the norms for the species; it is possible that an individual has fewer capabilities or a different structure (e.g. in the case that a limb has been lost).

- The map need not give values for the capabilities; rather just enumerate them. (Perhaps species averages may be expressed, though this is not required.)
User preferences are not taken into consideration at this stage, as per the principle of simplicity above. A separate higher-layer standard was developed for this.

By expressing profiles using OWL/RDF, each profile can link to a shared, standard, version of the capabilities and structure documents (and, indirectly, schema). This reduces information duplication and could promote adoption—and though it is not a key facet of this work from a research standpoint it is important to note that it assists adoption, which is a major goal.

The relationship between the above three components and the profiles is represented in Figure C.2.

The description and classification of capabilities can be cast as a small ontological modelling task. The ICF provides an internationally-recognised hierarchical method of classifying capabilities by gross structural area and capability specificity (i.e. more complex capabilities are derived from more general ones). The same approach was adopted for this work, but only a subset of ICF capabilities was described: many cognitive functions were omitted as this was not the focus of the work; also functions unrelated to the types of interfaces being studied were not included.
Appendix D

Adaptation Classification

This appendix presents a classification system that relates adaptations primarily by capability-centric, device-agnostic properties.

Each section of this appendix provides one facet of the overall classification—to classify a given instance of a given adaptation in a particular situation, one property value from each section/subsection should be selected.

D.1 Concrete Adaptation Classification

Concrete adaptations are the plug-ins and ATs that actually affect a change in the system on behalf of a person. In effect, they embody the syntactic outcome of any given abstract adaptation. Any concrete adaptation may be identified by its classification on a number of different criteria. This can be achieved by choosing one item from each subsection of this section.

D.1.1 Scope of Application

The area over which the adaptation is applied.

Abstract UI: mutation\(^1\) (widget choice and parameters).

A range of related mutations could be operating simultaneously, for different reasons, such as a preference for some types of folding, or colour types. Further: some current OSs and web browsers provide rudimentary support for basic mutations, such as a global minimum font size for widgets or content.\(^2\)

Device: modification.\(^3\) For a device class (e.g. all mice, keyboards or screens).

\(^1\)micro-adaptation \(\rightarrow\) \(\mu\)-adaptation \(\rightarrow\) mutation

\(^2\)Though as most GUI toolkits are not inherently adaptive, this is often expressed as an application- or system-level parameter that can be set by the user.

\(^3\)(macro-)adaptation \(\rightarrow\) modification
Table D.1: Adaptation scope-derived names.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Name</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>UI widgets</td>
<td>Mutation</td>
<td>Text size; TTS reading speed</td>
</tr>
<tr>
<td>Class of devices</td>
<td>Modification</td>
<td>Full-screen magnification; input filtering</td>
</tr>
<tr>
<td>Application</td>
<td>Adjustment</td>
<td>Word or Firefox zoom level</td>
</tr>
<tr>
<td>Content</td>
<td>Alteration</td>
<td>Filtering an RSS feed</td>
</tr>
</tbody>
</table>

**Application:** adjustment.\(^4\)

One further matter is of note, but out-of-scope for this work: that of content *alteration*.\(^5\) The direct manipulation of content is currently very uncommon—almost always the adaptation employed will interact with content via application-specific APIs (this includes cross-platform APIs such as the DOM, as such adaptations are currently almost always focussed on browser-displayed content adaptations). However, it is anticipated that in future there may be more of a focus on “direct” content adaptations apparently carried out outside of any given adaptation.

Though in reality there will always be some computer code interacting with the (stored representation of the) content, in future the blending of the web and Software as a Service (SaaS) into the dominant computing platform may make the distinction between the application and the content much harder to discern or even nonexistent as far as most users are concerned. In a future application that is hosted remotely and takes most of its data form different places on the Internet (a “mash up”), it could be very hard to discern which parts of the application originate form where and, thus, adjustments applied to “the application” would have to be applied to the content coming from multiple different sources.

This mandates a standard method for classifying and applying adaptations to served content. The current successful use of the DOM is likely to continue for some time, but more advanced (semantic) techniques may well need to be developed. Though it is out of the scope of the current work, the desired outcome of this work—to provide a more portable and generic method for specifying capability requirements—could act as the basis for such a system.

### D.1.2 Mechanisms

The method that the adaptation uses to affect a change.

**Parameters** are scalar values; e.g. display brightness (modification), widget colour (mutation) or minimum web page font size (adjustment).

\(^4\)name chosen for its implied limited scope

\(^5\)from the notion of altering (editing) content directly and also of alternative forms of content being provided for specific users, devices or situations
Filter take I/O and pre-/post-processes it.

Negotiation are like filters but intelligent and talks to the application/device about correcting its output rather than applying some generic post-processing.

Replacement of one device or channel with another. E.g. mouse → arrow keys. Also: using TTS to read some widgets.

D.1.3 Effects on System

The effects on the rest of the system (other channels, indicators) may also vary, as follows.

Mutations (adaptations applied to individual entities)

Requirements only: colour; widget style (e.g. menu items → buttons)

Requirements and indicators: reduction in content; folding; content substitutions.

Channel: size; font size; alternative content substitutions.

Modifications (adaptations applied to T-devices—and therefore all entities being rendered or input via the T-device)

Requirements only: screen brightness; loudspeaker volume.

Requirements and indicators: speech output speed; keyboard repeat rate; sticky keys; double-click speed.

Channel: full-screen magnification; switching entirely from visual to auditory or haptic output.

Adjustments (adaptations applied to applications; often settings/preferences within the particular program, affecting its behaviour)

Requirements only: switch locale or measurement units (improves user understanding).

Requirements and indicators: view settings (would also result in effective coarse mutations to the UI, such as changing the function or layout of different panes in the application’s window).

Channel: streaming video from the screen of one device to another.
Table D.2: Complete initiative-based classification of adaptive systems, based on Dieterich et. al.’s study [28]. (Reproduction of Table 2.1.)

<table>
<thead>
<tr>
<th>H Name</th>
<th>I</th>
<th>P</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Self-adaptation</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>• User-initiated self-adaptation</td>
<td>U</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>• Computer-aided adaptation</td>
<td>U</td>
<td>S</td>
<td>U</td>
<td>S</td>
</tr>
<tr>
<td>• User-controlled self-adaptation</td>
<td>S</td>
<td>S</td>
<td>U</td>
<td>S</td>
</tr>
<tr>
<td>• Adaptation</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>–</td>
</tr>
<tr>
<td>• System-initiated adaptation</td>
<td>S</td>
<td>U</td>
<td>U</td>
<td>–</td>
</tr>
<tr>
<td>◦ System-proposed, user-executed (unreasonable)</td>
<td>–</td>
<td>S</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>◦ User-proposed, system-decided (not used)</td>
<td>–</td>
<td>U</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>◦ System-decided, user-executed (unreasonable)</td>
<td>–</td>
<td>–</td>
<td>S</td>
<td>U</td>
</tr>
</tbody>
</table>

Notes: Emphasised names/characteristics come directly from the review. There are 16 possible systems as there are four stages of adaptation, each of which being carried out by (the user) XOR (the system). The table is grouped into blocks of four.

Fields: H (highlighted as interesting in paper): ● = yes; ○ = no. Stages: I = Initiative; P = Proposal; D = Decision; E = Execution. Actors: U = User; S = System; – = either (therefore accounting for two combinations).

D.2 Abstract Adaptation Classification

Abstract adaptations represent the semantic effects that may result from the application of any given concrete adaptation. There are only a small number of types of abstract adaptation and many matching concrete adaptations. Abstract adaptations have some effect on the flow of information between user and T-device, regardless of the modality of the channel or mechanism of the adaptation. These are introduced in subsection 6.6.3.

D.3 Behavioural Classification Revisited

The preceding sections may be used, together, to classify any type of abstract or more fully-specified concrete adaptation. To identify an instance of such an adaptation, one final piece of information may be added: the manner in which the adaptation instance was invoked.

Table D.2 is a copy of Table 2.1 for convenience and illustrates the possibilities for the four stages of adaptation: initiative; proposal; decision and execution.
D.4 Decisions about Adaptations

In relation to the discussion on how expressive adaptation feedback can be, in subsection 6.10.5, the following detailed examples are presented. Table D.3 enumerates the consequences for all possible abstract adaptations, in terms of TRIALS indicators and the user’s views on entity types and modalities. Table D.4 focuses on the implications for entity types and modalities when certain types of abstract adaptation are accepted or rejected.
Table D.3: Detailed abstract adaptation consequences.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>T</th>
<th>R</th>
<th>I</th>
<th>A</th>
<th>L</th>
<th>S</th>
<th>Type Modality</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A</strong></td>
<td>Bandwidth (channel)</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>modalit\textit{y}(channel) good</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
<td>(all in channel)</td>
</tr>
<tr>
<td></td>
<td>Reduce I requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>modalit\textit{y}(channel) good</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(all in channel)</td>
</tr>
<tr>
<td></td>
<td>Reduce E requirement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>modalit\textit{y}(channel) good</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(all in channel)</td>
</tr>
</tbody>
</table>

**Example scenarios:** * = swap images for alternative text; ** = swap visually-rendered text for TTS; *** = swap images for text read with TTS; † = rendering text via Braille and TTS; †† = listening to and visualising music.
### Table D.4: Accept and reject action implications.

<table>
<thead>
<tr>
<th>Accept</th>
<th>Adaptation</th>
<th>Type</th>
<th>Modality</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$P \to {P, A}$</td>
<td>(whole channel)</td>
<td>$\text{modality}(\text{channel})$ good</td>
</tr>
<tr>
<td>B</td>
<td>$P \to {P, B}$</td>
<td>atype(e) good</td>
<td>$\text{modality}(e)$ good</td>
</tr>
<tr>
<td>RT</td>
<td>$P \to {P, RT}$</td>
<td>atype(e) $\to$ atype(e')</td>
<td>$\text{modality}(e)$ good $\to$ $\text{modality}(e')$</td>
</tr>
<tr>
<td>RM</td>
<td>$P \to {P, RM}$</td>
<td>atype(e) good</td>
<td>$\text{modality}(e)$ $\to$ $\text{modality}(e')$</td>
</tr>
<tr>
<td>R+</td>
<td>$P \to {P, R+}$</td>
<td>atype(e) $\to$ atype(e')</td>
<td>$\text{modality}(e)$ $\to$ $\text{modality}(e')$</td>
</tr>
<tr>
<td>SM</td>
<td>$P \to {P, SM}$</td>
<td>atype(e) good</td>
<td>$\text{modality}(e)$ $\to$ ${\text{modality}(e), \text{modality}(e')}$</td>
</tr>
<tr>
<td>S+</td>
<td>$P \to {P, S+}$</td>
<td>atype(e) $\to$ ${\text{atype}(e), \text{atype}(e')}$</td>
<td>$\text{modality}(e)$ $\to$ ${\text{modality}(e), \text{modality}(e')}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reject</th>
<th>Adaptation</th>
<th>Type</th>
<th>Modality</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$P \leftarrow {P, A}$</td>
<td>(whole channel)</td>
<td>$\text{modality}(\text{channel})$ good</td>
</tr>
<tr>
<td>B</td>
<td>$P \leftarrow {P, B}$</td>
<td>atype(e) good</td>
<td>$\text{modality}(e)$ good</td>
</tr>
<tr>
<td>RT</td>
<td>$P \leftarrow {P, RT}$</td>
<td>atype(e) $\leftarrow$ atype(e')</td>
<td>$\text{modality}(e)$ good</td>
</tr>
<tr>
<td>RM</td>
<td>$P \leftarrow {P, RM}$</td>
<td>atype(e) good</td>
<td>$\text{modality}(e)$ $\leftarrow$ $\text{modality}(e')$</td>
</tr>
<tr>
<td>R+</td>
<td>$P \leftarrow {P, R+}$</td>
<td>atype(e) $\leftarrow$ atype(e')</td>
<td>$\text{modality}(e)$ $\leftarrow$ $\text{modality}(e')$</td>
</tr>
<tr>
<td>SM</td>
<td>$P \leftarrow {P, SM}$</td>
<td>atype(e) good</td>
<td>$\text{modality}(e)$ $\leftarrow$ ${\text{modality}(e), \text{modality}(e')}$</td>
</tr>
<tr>
<td>S+</td>
<td>$P \leftarrow {P, S+}$</td>
<td>atype(e) $\leftarrow$ ${\text{atype}(e), \text{atype}(e')}$</td>
<td>$\text{modality}(e)$ $\leftarrow$ ${\text{modality}(e), \text{modality}(e')}$</td>
</tr>
</tbody>
</table>

**Notes:** preceding set of adaptations is denoted $P$; abstract entity (or group thereof) to which an adaptation applies is denoted $e$; modified entity, following adaptation is denoted $e'$. 
Appendix E

Example Adaptation–Capability Mapping

This is the hand-crafted adaptation–capability mapping used in tests (section 8.4).

ScreenMag
  VisualAcuity
Colours
  LearningStyle
  ColourPerception
  ContrastSensitivity
FoldUI
  Stamina
  FineControl
FontSize
  VisualAcuity
SimpleUI
  ShortTermMemoryLevel
  GrossControl
  FineControl
  VisualAcuity
  ReadingAge
Summarise
  VisualStamina
  Stamina
  ReadingAge
  ShortTermMemoryLevel
OpponentArrow
  PositionalIdentification
APPENDIX E. EXAMPLE ADAPTATION–CAPABILITY MAPPING

AmplitudeRange
FrequencyRange
MouseForTrackball
   Flexion
   FineControl
TTS
   VisualAcuity
   ReadingAge
   LearningStyle
   LightSensitivity
MouseForKeyboard
   FineControl
   Stamina
   Flexion
BiggerButtons
   GrossControl
   Stamina
DoubleClick
   FineControl
   Stamina
   Strength
PageZoom
   VisualAcuity
Subtitles
   VoiceDiscernment
   CrowdVoiceDiscernment
   AmplitudeRange
   FrequencyRange
Appendix F

Publications and Seminars

This appendix details the publications and other outputs and activities resulting from work towards this thesis. Publications were peer-reviewed except when stated otherwise.

F.1 Conference Papers

F.1.1 Adaptive Accessibility

• Towards ubiquitous accessibility: capability-based profiles and adaptations, delivered via the semantic web. W4A 2012, Lyon [6].

• Towards accessible interactions with pervasive interfaces, based on human capabilities. ICCHP 2010, Vienna [12] (extended abstract reviewed).

• The Potential of Adaptive Interfaces as an Accessibility Aid for Older Web Users. W4A 2010, Raleigh [101].


F.1.2 Web and Document Accessibility

• (See also “Towards ubiquitous accessibility: capability-based profiles and adaptations, delivered via the semantic web” above.)

• Opening up Access to Online Documents using Essentiality Tracks. W4A at WWW 2006, Edinburgh [2].
F.1.3 Game Accessibility


- Making the Mainstream Accessible: Redefining the Game, Sandbox Symposium at SIGGRAPH 2006, Boston [4].


- Making the Mainstream Accessible: More than a Game, Fun ’n Games 2006, Preston [3].

F.2 Workshop Papers

F.2.1 User Modelling

- Modelling of Users’ Capabilities. IUI4AAL at IUI 2008, Gran Canaria [9].

- DSAI\textsuperscript{1} 2007 [8] (invited speaker).

F.3 Book Chapters

- Chapter in “Improving Library Services for People with Disabilities” [7] (co-authored with Colin Machin and Jatinder Dhiensa).

F.4 Held Workshops and Seminars

F.4.1 External

- November 2005: Accessible Games Workshop, Johannes Kepler Universität, Linz (invited speaker).


\textsuperscript{1}http://dsai2007.utad.pt/
F.4.2 Internal

- “AudioQuake”—technical discussion on making games accessible for the Student Union’s Computer Society. This event won an award from the Computer Society at its AGM.

- “Making the Mainstream Accessible”—talk to Knowledge Management research group.

- “Opening up Access to Online Documents using Essentiality Tracks”—seminar to Knowledge Management research group.

- “Formal User Modelling”—talk to Knowledge Management research group.

F.4.3 Workshops for Research Students

- Paper Writing (7th March 2008), with thanks to Ray Dawson.

- Paper Presenting (14th March 2008).